J Glob Optim (2013) 56:1101–1121
DOI 10.1007/s10898-012-9928-x

On interval branch-and-bound for additively separable functions with common variables

J. L. Berenguel · L. G. Casado · I. García · E. M. T. Hendrix · F. Messine

Received: 14 November 2011 / Accepted: 17 May 2012 / Published online: 4 June 2012

Abstract Interval branch-and-bound (B&B) algorithms are powerful methods which look for guaranteed solutions of global optimisation problems. The computational effort needed to reach this aim, increases exponentially with the problem dimension in the worst case. For separable functions this effort is less, as lower dimensional sub-problems can be solved individually. The question is how to design specific methods for cases where the objective function can be considered separable, but common variables occur in the sub-problems. This paper is devoted to establish the bases of B&B algorithms for separable problems. New B&B rules are presented based on derived properties to compute bounds. A numerical

This work has been funded by grants from the Spanish Ministry of Science and Innovation (TIN2008-01117), Junta de Andalucía (P08-TIC3518 and P11-TIC7176), in part financed by the European Regional Development Fund (ERDF) and Seneca Foundation (Murcia Region nr 15254/PI/10). Eligius Hendrix is a fellow of the Spanish "Ramon y Cajal" contract program, co-financed by the European Social Fund.

J. L. Berenguel
TIC 146: "Supercomputing-Algorithms" Research Group, University of Almería,
Agrifood Campus of International Excellence (ceiA3), 04120 Almería, Spain
e-mail: jlberenguel@gmail.com

L. G. Casado (✉)
Department of Computer Architecture and Electronics, University of Almería, 04120 Almería, Spain
e-mail: leo@ual.es

I. García · E. M. T. Hendrix
Department of Computer Architecture, University of Málaga, Campus de Teatinos, 29017 Málaga, Spain
e-mail: igarciaf@uma.es

E. M. T. Hendrix
e-mail: Eligius@uma.es

E. M. T. Hendrix
Operations Research and Logistics, Wageningen University, Wageningen, The Netherlands

F. Messine
ENSEEIHT-IRIT UMR-CNRS-5505, University of Toulouse, 2 rue Camichel, 31000 Toulouse, France
e-mail: Frederic.Messine@n7.fr

illustration is elaborated with a test-bed of problems mostly generated by combining traditional box constrained global optimisation problems, to show the potential of using the derived theoretical basis.

Keywords Branch-and-bound · Interval arithmetic · Separable functions

1 Introduction

Interval branch-and-bound methods look for guaranteed solutions of global optimisation problems [4,6,7,12,15]. Although these methods have the ability to handle constraints, we focus here on the generic box constrained global optimisation problem, which is to find

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

where $S \in \mathbb{I}^n$ is the search region and $\mathbb{I}$ stands for the set of all closed real intervals. With increasing dimension n, the computational effort increases drastically. In design problems, it is not unusual that the objective function f is the composition of several sub-functions, as for example to design electrical devices [9]. If this is the case in an additive way and the variables can be split into subgroups that do not overlap, we call the function completely additively separable.

To be precise, we introduce the following notation. The complete search space is $S = (S_1, \ldots, S_n) \in \mathbb{I}^n$ and the index set of variables is $I = \{1, \ldots, n\}$. Index sets $I^{[j]} = \{i_1, \ldots, i_{n^{[j]}}\} \subseteq I$ with $n^{[j]} = |I^{[j]}|$ elements are used to denote subgroups of variables and their corresponding search region $S^{[j]} \in \mathbb{I}^{n^{[j]}}$. In this context, we use the following definitions.

Definition 1 Function $f : S \subset \mathbb{R}^n \to \mathbb{R}$ is additively separable, if $\exists p > 1$ such that f can be written as

$$f(x) = \sum_{j=1}^{p} f^{[j]}(x^{[j]}),\ x^{[j]} \in S^{[j]}. \tag{2}$$

The index sets facilitate the definition of a completely separable function.

Definition 2 Function $f : S \subset \mathbb{R}^n \to \mathbb{R}$ is called completely separable, if it can be written as (2) and $I^{[j]} \cap I^{[k]} = \emptyset,\ \forall k, j \in \{1, \ldots, p\}, j \neq k$.

We summarise the notation used in the mathematical description:

- Box, or n-dimensional interval: $X = (X_1, \ldots, X_i = [\underline{x}_i, \overline{x}_i], \ldots, X_n)$.
- $m(X)$ is the midpoint of the box X.
- $w(X)$ is the width of X, i.e., the width of the largest component of X.
- n-dimensional point: $x = (x_1, \ldots, x_n)^T$.
- Sub-function: $f^{[j]},\ j \in \{1, \ldots, p\}$.
- Subbox: $X^{[j]} \subseteq S^{[j]}$.
- Subpoint: $x^{[j]} \in S^{[j]} \subseteq \mathbb{R}^{n^{[j]}}$.

Problem (1) can be solved by considering the addition of subproblems, for completely separable functions:

$$\min_{x \in S} f(x) = \sum_{j=1}^{p} \min_{x^{[j]} \in S^{[j]}} f^{[j]}(x^{[j]}). \tag{3}$$

In general, problem (3) is easier to solve than (1), as the subproblems have a lower dimension. However, in engineering design, cases may have an additive structure in the objective function, but common variables are shared by the underlying functions, see [9]. Nocedal and Wright [13] mention in their book the concept of so-called partially separable problems to derive numerical advantages in Quasi-Newton methods if the Hessian is sparse.

We focus on the case where one group of variables appears in each sub-function. The developed theory is valid for more than one common variable, but in order to keep notation comprehensible, we present it for one common variable. This theory presents the basis for future studies on more complicated relations in the model. Without loss of generality, the common variable is denoted by the last one, x_n. So, common variable x_n appears in all sub-functions, $n \in I^{[1]} \cap \ldots \cap I^{[p]}$. Moreover, in the numerical examples we restrict ourselves to $p = 2$.

To separate the common variable x_n from the non-common variables, the latter will be denoted by $z^{[j]} \in T^{[j]}$ such that $S^{[j]} = (T^{[j]}, S_n)$. The generic problem under investigation can be written as

$$\min_{z^{[j]} \in T^{[j]}, x_n \in S_n} \left\{ f(x) = \sum_{j=1}^{p} f^{[j]}(z^{[j]}, x_n) \right\}, \tag{4}$$

where each vector $x^{[j]} = (z^{[j]}, x_n) \in \mathbb{R}^{n^{[j]}}$ should fulfil individual box constraints $x^{[j]} \in S^{[j]} \in \mathbb{I}^{n^{[j]}}$ and the common variable $x_n \in S_n \in \mathbb{I}$. There are several ways to denote the common variable, depending on its context. The common variable are denoted by x_n when $x \in S$ and $x_{n^{[j]}}$ when $x \in S^{[j]}$.

Example 1 Consider the function $f(x) = x_1^2 + x_1x_3 - 2x_2x_3 + x_2^2 + \frac{1}{2}x_3^2 + x_3$. We can set $z^{[1]} = x_1$ and $z^{[2]} = x_2$, such that $x^{[1]} = (z^{[1]}, x_3)$ and $x^{[2]} = (z^{[2]}, x_3)$. In this way function f can be rewritten as

$$f(x) = f^{[1]}(x^{[1]}) + f^{[2]}(x^{[2]}),$$

where

$$f^{[1]}(z^{[1]}, x_3) = x_1^2 + x_1x_3 + \frac{1}{2}x_3^2 + x_3 = (z_1^{[1]})^2 + z_1^{[1]}x_3 + \frac{1}{2}x_3^2 + x_3$$

and

$$f^{[2]}(z^{[2]}, x_3) = x_2^2 - 2x_2x_3 = (z_1^{[2]})^2 - 2z_1^{[2]}x_3.$$

Section 2 summarizes interval arithmetic properties relevant for this study. Section 3 shows a standard interval branch-and-bound algorithm which is used for comparison. Section 4 presents the so-called decomposed sub-function perspective (DSP). Section 5 studies DSP properties. This leads to a specific branch-and-bound algorithm described in Sect. 6. Finally, a numerical illustration and conclusions are presented in Sects. 7 and 8, respectively.

2 Properties of interval inclusion functions

Algorithms based on interval arithmetic have several ingredients. We start with the generic ideas and then focus on how they can be adapted to separable functions with a common variable. Interval arithmetic has been widely studied in the last 40 years [10,11]. We mention some relevant interval arithmetic properties and definitions.

Definition 3 Let $f(X)$ be the range of f on X. An interval function $F : \mathbb{I}^n \to \mathbb{I}$ is an *inclusion function* if and only if $f(X) \subseteq F(X) = [\underline{F}(X), \overline{F}(X)]$.

Definition 4 Inclusion isotonicity. Inclusion function $F : \mathbb{I}^n \to \mathbb{I}$ of f is *inclusion isotone*, if and only if $\forall (X, Y) \in \mathbb{I}^n$, $X \subseteq Y \Rightarrow F(X) \subseteq F(Y)$.

In the following, we assume that F is an isotone inclusion function.

Property 1 Refinement of F over X. If $\bigcup_{k=1}^{d} X^k$ is a partitioning of X then $F(X) \supseteq \bigcup_{k=1}^{d} F(X^k)$, according to Definition 4.

Property 2 Refinement of $\underline{F}$ over X. If $\bigcup_{k=1}^{d} X^k$ is a partitioning of X then $\min_k\{\underline{F}(X^k)\} \geq \underline{F}(X)$ is a valid lower bound of f over X. This follows directly from Property 1 and is of interest for minimisation problems.

In interval analysis [10], isotone inclusion functions can be constructed as follows:

Property 3 Fundamental Theorem of Interval Analysis. Given a real function $f : \mathbb{R}^n \to \mathbb{R}$, a natural interval extension of f, denoted by F_{NIE}, is an isotone inclusion function [10].

In this work, we also use the Baumann inclusion function F_B based on the first order Taylor expansion [1,16]:

$$F_B(X) = F_{NIE}(c) + F'_{NIE}(X) \cdot (X - c) = F_{NIE}(c) + \sum_{i=1}^{n} (F'_{NIE})_i(X)(X_i - c_i),$$

where $c \in X$ and $F'_{NIE}(X)$ is the component-wise enclosure of the gradient. In [1], Baumann proved that the best lower bound using this formulation is generated choosing

- If $(F'_{NIE})_i(X) < 0$, $c_i = \overline{x}_i$.
- If $(F'_{NIE})_i(X) > 0$, $c_i = \underline{x}_i$.
- If $0 \in (F'_{NIE})_i(X)$, $c_i = \frac{\underline{x}_i(\overline{F'}_{NIE})_i(X) - \overline{x}_i(\underline{F'}_{NIE})_i(X)}{(\overline{F'}_{NIE})_i(X) - (\underline{F'}_{NIE})_i(X)}$.

The intersection of an isotone inclusion function with another inclusion function also provides an isotone inclusion function:

$$F_{\cap}(X) := F_{NIE}(X) \cap F_B(X), \tag{5}$$

Other isotone inclusion functions are described in [5,8,14,16,17].

Property 4 Monotonicity. Let $(F'_{NIE})_i$ be an interval inclusion of the partial derivative of f with respect to x_i. If $\exists i \;\; 0 \notin (F'_{NIE})_i(X)$, then the interval X does not contain a stationary point.

Property 4 is useful to check whether a differentiable function may have an optimum point in the interior of a box X.

3 Standard interval branch-and-bound algorithm (IBB)

The properties above are used in standard interval global optimisation algorithms (see Algorithm 1), e.g. [16]. Several branch-and-bound rules are described here. In Sect. 5 we deal with extending them for separable functions with common variables.

Algorithm 1 : General interval B&B algorithm

Funct IBB(S, f)

1. Set the working list $L := \{S\}$ and the final list $Q := \emptyset$
2. Set $f^U = \overline{F}(m(S))$
3. **while** ($L \neq \emptyset$)
4. Select an interval X from L *Selection rule*
5. Compute $\overline{F}(m(X))$
6. **if** $\overline{F}(m(X)) < f^U$
7. $f^U = \overline{F}(m(X))$
8. Remove all $X \in L \cup Q$ with $\underline{F}(X) > f^U$ *Cut-off test*
9. Divide X into subintervals $X^j,\ j = 1, \ldots, k$ *Division rule*
10. **for** $j = 1$ to k
11. Compute a lower bound $\underline{F}(X^j)$ of $f(X^j)$ *Bounding rule*
12. **if** X^j cannot be eliminated *Elimination rule*
13. **if** X^j satisfies the termination criterion *Termination rule*
14. Store X^j in Q
15. **else**
16. Store X^j in L
17. **return** Q

- Bounding: $\underline{F}(X)$ is a lower bound of f on X. Upper bound f^U of f^* is the lowest function value found in the midpoint $m(X)$ of the evaluated boxes X, $\overline{F}(m(X))$.
- Selection: box X with the lowest value for bound $\underline{F}(X)$ is selected (best first search) in order to find improvements of upper bound f^U of f^* in an early stage.
- Elimination:
  - RangeUp test: X is rejected if $\underline{F}(X) > f^U$.
  - Monotonicity test: box X interior to S is rejected if $0 \notin F'(X)$.
- Division: Selected and not eliminated interval X is bisected by the midpoint of the widest component. This avoids slicing off small parts assuring convergence of the algorithm.
- Termination: If one intends to find all minimum points, a common termination rule is to finish when the size of all boxes is sufficiently small. Algorithm 1 stores boxes with $w(X) \leq \epsilon$ in the final list Q.

As a result of the algorithm, the set of minimum points x^* is enclosed by $\cup_{X \in Q} X$, and f^* is enclosed by the interval $[\min_{X \in Q} \underline{F}(X),\ f^U]$.

4 Decomposed sub-function perspective (DSP)

The basic question is how to make use of the fact that the function is separable with common variables when constructing a specific B&B algorithm. One would like to make use of the fact that the subproblems have a dimension $n^{[j]}$ smaller than n.

One way is to consider copies $X_n^{[j]}$ of subboxes $X_n \subseteq S_n$ for each sub-function. In a branch-and-bound environment, this implies keeping lists $L^{[j]}$ and $Q^{[j]}$ for each sub-function $f^{[j]}$ in an $n^{[j]}$-dimensional space.

Example 2 Consider Example 1 from the decomposed sub-function perspective. We have copies $x_3^{[1]}, x_3^{[2]}$ of the variable x_3. Relaxing the constraint $x_3^{[1]} = x_3^{[2]}$ gives a lower bound of f over $[-10, 10]^3$. Minimising $f^{[1]}(z^{[1]}, x_3^{[1]})$ gives a unique minimum point $(z^{[1]}, x_3^{[1]})$

$= (1, -2)$ with function value $f^{[1]}(z^{[1]}, x_3^{[1]}) = -1$. Similarly $f^{[2]}(z^{[2]}, x_3^{[2]})$ has a minimum of -100 attained at the vertices $(-10, -10)$ and $(10, 10)$, such that the resulting global lower bound is -101.

Instead of only one list L in Algorithm 1, we have a set of p lists $L^{[j]}$ to store the $n^{[j]}$-dimensional (generated and not eliminated) subproblems for each $f^{[j]}$ sub-function; $L^{[j]} = \{X = (Z^{[j]}, X_{n^{[j]}})\}$ with lower bounds $\underline{F}^{[j]}(X)$. The algorithm finishes when all $L^{[j]}$ are empty. Final boxes are stored in $Q^{[j]}$. For each sub-function j, the lower bound on the sub-function is given by

$$lb^{[j]} = \min_{X \in L^{[j]} \cup Q^{[j]}} \underline{F}^{[j]}(X). \tag{6}$$

Notice that after running the algorithm, these lists have to be combined again into a final list Q in n-dimensional space. This requires a last phase to combine those subboxes that have the same range of the common variable.

For higher dimension, we expect to have two effects of using a decomposed approach compared to the standard (full dimensional) Algorithm 1. First of all, the number of boxes generated by Algorithm 1 in the worst case has an exponential behaviour in the dimension. Moreover, in higher dimension the overestimation of interval arithmetic can be larger than in lower dimension when there exist multiple occurrences of variables. If the ratio between non-common and common variables is large, using a decomposition may provide more advantage.

In the following section, we introduce properties of a separable function with a common variable that can be used for the decomposed sub-function perspective (DSP). These properties are the basis for a specific branch-and-bound algorithm.

5 Properties of a separable function with a common variable

We focus on properties that are of interest in the DSP approach in order to avoid searching in areas not containing a global solution. For this purpose, one should obtain bounds of function values as sharp as possible.

5.1 Bounding

We have to distinguish bounds of $f^{[j]}$ from bounds of f. A lower bound of $f^{[j]}$ is relatively easy to calculate using inclusion functions $F^{[j]}(X \subseteq S^{[j]})$, see Property 3.

Property 5 By Property 2, $lb^{[j]}$ is a lower bound of $f^{[j]}$ over $S^{[j]}$.

An upper bound of the minimum of $f^{[j]}$ on $X \subseteq S^{[j]}$ can now be defined as depending on the value of the common variable x_n at the midpoint $x_n = m(X_{n^{[j]}})$. Let $g^{[j]}(x_n)$ be the best value found so far of $\overline{F}^{[j]}(m(X))$ for an $X \in L^{[j]} \cup Q^{[j]}$ with the common variable having a value of $m(X_{n^{[j]}}) = x_n$. Extension of the general bounding concept to the sub-functions is then straightforward as formalised in the following theorem.

Theorem 1 *Let interval $X \in S^{[j]}$. If $\underline{F}^{[j]}(X) > g^{[j]}(x_n)$ with $x_n \in X_{n^{[j]}}$, then X cannot contain a $n^{[j]}$-dimensional part of an optimum point x^*.*

There are several ways to obtain a lower bound of f. A straightforward observation is the following.

Property 6 By Definition 1, $\sum_{j=1}^{p} lb^{[j]}$ is a lower bound of f over S.

Function f is defined in the n-dimensional space, so a bound of f on $X \subseteq S^{[j]}$ is not defined. Alternatively, one can look for the lowest values that f can reach when box X marks the solution in the $n^{[j]}$-dimensional subspace. For this, it is convenient to define for each interval $Y \subset S_n$ in the common variable space and for each sub-function j

$$\Psi^{[j]}(Y) = \arg\min\{\underline{F}^{[j]}(X) \mid X \in L^{[j]} \cup Q^{[j]},\ X_{n^{[j]}} \cap Y \neq \emptyset\} \tag{7}$$

as the subbox in the set of non-rejected subboxes in lists $L^{[j]} \cup Q^{[j]}$ with the smallest lower bound of $f^{[j]}$ and where the range of the common variable overlaps with Y. The following proposition will facilitate the reasoning.

Proposition 1 *Given an interval Y of S_n obtained by an iterative uniform division, $\underline{F}^{[j]}(\Psi^{[j]}(Y) = (Z, X_{n^{[j]}}))$ is a lower bound of $f^{[j]}$ over $(T^{[j]}, Y)$.*

Proof To obtain a lower bound of $f^{[j]}$ over $(T^{[j]}, Y)$, we focus on $\Psi^{[j]}(Y)$ which has the smallest $\underline{F}^{[j]}(Z, X_{n^{[j]}})$ with $X_{n^{[j]}} \cap Y \neq \emptyset$. Consider all cases:

- $Y \subseteq X_{n^{[j]}}$, the result follows from the definition of $\Psi^{[j]}(Y)$ and inclusion isotonicity of interval arithmetic (see Definition 4).
- $X_{n^{[j]}} \subset Y$. Because Y and $X_{n^{[j]}}$ are generated from the same division scheme, $X_{n^{[j]}}$ belongs to a uniform subdivision of Y. According to Property 2, $\underline{F}(Z, X_{n^{[j]}})$ is a correct lower bound of $f^{[j]}$ over $(T^{[j]}, Y)$.
- $X_{n^{[j]}} \cap Y \neq \emptyset, Y \not\subseteq X_{n^{[j]}}$ and $X_{n^{[j]}} \not\subset Y$ cannot happen because all intervals are generated using the same division scheme.

□

Given a subbox $X \in S^{[j]}$, a corresponding lower bound of f can be obtained as follows. Consider f to be evaluated over a (extended) box with the components of $I^{[j]}$ limited to the subbox X and the other $I \setminus I^{[j]}$ components are free in S.

Definition 5 Given a subbox $X \in S^{[j]}$, its extension to S is defined as $EX^{[j]} = \{E \in \mathbb{I}^n \mid E \subseteq S,\ E^{[j]} = X\}$ or in another way $EX^{[j]} = \{x \in S \mid x_m \in X_m,\ m \in I^{[j]};\ x_l \in S_l,\ l \notin I^{[j]}\}$.

Proposition 2 *Consider sub-function j and a corresponding box $X \in L^{[j]} \cup Q^{[j]}$. A lower bound of f over $EX^{[j]}$ is given by*

$$\underline{F}^{[j]}(X) + \sum_{k=1, k\neq j}^{p} lb^{[k]}. \tag{8}$$

Proof Based on (6), Property 5 and inclusion isotonicity (Definition 4),

$$lb^{[k]} = \min_{X \in L^{[k]} \cup Q^{[k]}} \underline{F}^{[k]}(X) \leq f^{[k]}(T^{[k]}, X_{n^{[j]}}). \tag{9}$$

So,

$$\underline{F}^{[j]}(X) + \sum_{k=1, k\neq j}^{p} lb^{[k]} \leq f^{[j]}(X) + \sum_{k=1, k\neq j}^{p} f^{[k]}(T^{[k]}, X_{n^{[j]}}) = f(EX^{[j]}). \tag{10}$$

□

A sharper lower bound of f over $(EX^{[j]})$ is given by the following theorem.

Theorem 2 *Consider a subbox $X \in L^{[j]} \cup Q^{[j]}$.*

$$\underline{F}_{DSP}(EX^{[j]}) := \underline{F}^{[j]}(X) + \sum_{k=1,k\neq j}^{p} \underline{F}^{[k]}(\Psi^{[k]}(X_{n^{[j]}})) \tag{11}$$

is a valid lower bound of f over $EX^{[j]}$.

Proof This theorem is a direct consequence of Propositions 1 and 2. □

Property 7 Subbox $X \in L^{[j]} \cup Q^{[j]}$ can be removed if $\exists k \neq j,\ \forall Y \in L^{[k]} \cup Q^{[k]},\ X_{n^{[j]}} \cap Y_{n^{[k]}} = \emptyset$.

Upper bound f^U of f^* is usually updated by evaluating f at a point x. The question is how to select x for an interval $X = (Z, X_{n^{[j]}}) \subset S^{[j]} \neq S$. The components of the non-common variables can be taken as the midpoint of Z and the non-common part of the other sub-functions from the corresponding $\Psi^{[k]}(X_{n^{[j]}}),\ k \neq j$. For the value of the common variable x_n, we should look in the overlapping (intersection) part of X and $\Psi^{[k]}(X_{n^{[j]}})$.

5.2 Monotonicity

The monotonicity property for non-common variables is straightforward and is valid for each list separately.

Property 8 Non-common variables monotonicity. Property 4 applies to non-common variables.

Property 9 Common variables monotonicity. From Property 4, focusing on the common variable x_n, we know that for an interior optimum

$$\frac{\partial f}{\partial x_n} = \sum_j \frac{\partial f^{[j]}}{\partial x_n} = 0. \tag{12}$$

Similar to the bounding rule, one has to keep track of what happens in the other lists when checking monotonicity in the common variable. We introduce the following notation for the bound on the derivative:

$$\underline{\Theta}^{[j]}(Y) = \min\left\{\underline{F'}^{[j]}_{n^{[j]}}(X) \mid X \in L^{[j]} \cup Q^{[j]},\ X_{n^{[j]}} \cap Y \neq \emptyset\right\} \tag{13}$$

and

$$\overline{\Theta}^{[j]}(Y) = \max\left\{\overline{F}'^{[j]}_{n^{[j]}}(X) \mid X \in L^{[j]} \cup Q^{[j]},\ X_{n^{[j]}} \cap Y \neq \emptyset\right\}. \tag{14}$$

Now we can formulate the monotonicity property for the common variable using the following theorem.

Theorem 3 *Consider subbox $X \subset S^{[j]}$. X cannot contain an $n^{[j]}$-dimensional part of an interior optimum point x^* if*

$$\underline{F}'^{[j]}_{n^{[j]}}(X) + \sum_{k=1,k\neq j}^{p} \underline{\Theta}^{[k]}(X_{n^{[j]}}) > 0 \tag{15}$$

or

$$\overline{F}^{[\prime j]}_{n^{[j]}}(X) + \sum_{k=1,k\neq j}^{p} \overline{\Theta}^{[k]}(X_{n^{[j]}}) < 0. \tag{16}$$

Proof This theorem follows from considering (12) taking into account that either $\sum_j \frac{\partial f^{[j]}}{\partial x_n} < 0$, or $\sum_j \frac{\partial f^{[j]}}{\partial x_n} > 0$. □

6 DSP based interval branch-and-bound algorithm (IBB–DSP)

The ingredients of the sketched IBB algorithm are extended towards the DSP approach.

6.1 Bounding rule

A lower bound of $f^{[i]}(X)$ is obtained by Eq. (6) and a lower bound of $f(X)$ is obtained by Eq. (11). A list of upper bounds $g^{[j]}(x_n)$ is maintained for those values x_n such that $\exists Y \in L^{[j]} \cup Q^{[j]}$ with $m(Y_{n^{[j]}}) = x_n$.

6.2 Selection rule

This rule determines the subproblems to be visited in the search tree, searching for the most promising regions. First it should be decided which list $L^{[j]}$ to process next. A round robin scheme can be applied in the set of non-empty lists, but other schemes can also be considered. For the selection of the subbox X from a list $L^{[j]}$ to be processed next we evaluated the following criteria.

1. subbox X with the lowest value for bound $\underline{F}^{[j]}(X)$.
2. subbox X with the largest width $w(X)$.
3. subbox X with lowest saved bound of extension $\underline{F}_{DSP}(EX^{[j]})$.
4. subbox X with the widest range for $X_{n^{[j]}}$ and as second rule having the lowest extended bound $\underline{F}_{DSP}(EX^{[j]})$.
5. Take X according to 3. If $\exists k \neq j$ with $w(\Psi^{[k]}(X_{n^{[j]}})) > w(X)$, select $\Psi^{[k]}(X_{n^{[j]}})$ instead.
6. Take X according to 3. Evaluate $\underline{F}_{DSP}(EX^{[j]})$. If its value is higher than the saved value, restore X and repeat 3.

Criterion 1 does usually not improve the upper bound of f^* due to its focus on subproblem bounds instead of the full objective function.

Criterion 2 is a Breadth-First selection rule which attempts to improve values of $\underline{F}_{DSP}(EX^{[j]})$ (see (11)) by diminishing the width of boxes in the search tree, avoiding large differences in the size of the boxes in both lists.

Criterion 3 seems a good choice because it uses Best-First and global information, similar to the IBB algorithm. However, each subproblem usually has a different search behaviour. They differ in the total number of evaluated subproblems and in the number of evaluated subproblems per search tree level. So, it is difficult to select the box that improves $\underline{F}_{DSP}(EX^{[j]})$ in the future, because $\underline{F}_{DSP}(EX^{[j]})$ depends not only on X, but also on $\Psi^{[k]}(X_{n^{[j]}})$ (see (7) and (11)) which are updated during the algorithm run.

Criterion 4 combines 2 and 3 with their advantages and drawbacks.

Criterion 5 attempts to improve the value of $\underline{F}_{DSP}(EX^{[j]})$ by dividing the widest box among $X^{[j]}$ and $\Psi^{[k]}(X_n^{[j]})$.

Criterion 6 tries to maintain the values of $\underline{F}_{DSP}(EX^{[j]})$ updated. According to our preliminary experiments, this is the best of the presented selection rules and it is used in the illustration in Section 7.

6.3 Elimination rules

The following rules can be used to eliminate subboxes $X = (Z, X_{n^{[j]}}) \in L^{[j]} \cup Q^{[j]}$ that are shown not to contain components of the optimal solution.
RangeUp test (RUpT): X is rejected if $\underline{F}_{DSP}(EX^{[j]}) > f^U$.
Sub-function RangeUp Test (SubRUpT): Remove X if $\underline{F}^{[j]}(X) > g^{[j]}(x_n)$ with $x_n \in X_{n^{[j]}}$ (Theorem 1).
Unshared Value of Common Variable (UVCV): Box X is rejected if $\exists k \neq j\ \Psi^{[k]}(X_{n^{[j]}}) = \emptyset$ (Property 7).
Non-Common Variables Monotonicity Test (NCVMT): X is rejected, if $0 \notin F_i^{[rj]}(X),\ i \neq n^{[j]}$ and Z is interior with respect to $T^{[j]}$ (Property 8).

6.4 Division rule

Similar to the IBB algorithm, the selected and not eliminated subbox $X \in S^{[j]}$ is bisected by the midpoint of the widest component.

6.5 Termination rule

All subboxes $X \in S^{[j]}$ with $w(X) \leq \epsilon$ are stored in the final list $Q^{[j]}$. The algorithm combines in the end $Q^{[i]},\ i = 1, \ldots, p$, lists into a final list Q of boxes in n-dimensional space. Due to the division rule, the algorithm has a finite number of steps. As subboxes that are guaranteed not to contain components of the optimal solution are thrown out, the algorithm converges to a finite set of combinations that contains the optimal solutions.

6.6 Implementation aspects

One of the most time consuming parts of the algorithm is the search of $\Psi^{[k]}(X_{n^{[j]}})$ to get $\underline{F}_{DSP}(EX^{[j]})$ in Eq. (11). To avoid the search in the $L^{[j]}$ and $Q^{[j]}$ data structures, that usually contain many elements, an auxiliary data structure $M^{[j]}$ is created. $M^{[j]}$ is a list of $\Psi^{[j]}(X_{n^{[j]}})$ for all existing intervals $X_{n^{[j]}}$.
The IBB–DSP algorithm has three phases:

- Phase 1: Running a B&B algorithm based on the described rules resulting in lists of final boxes $Q^{[j]}$.
- Phase 2: The value of $\underline{F}_{DSP}(EX^{[j]})$ is updated for all $X \in Q^{[j]}$. The current value of $\underline{F}_{DSP}(EX^{[j]})$ can be based on a stored $\Psi^{[k]}(X_{n^{[j]}})$. The aim is the elimination of boxes using the RUpT and UVCV tests.
- Phase 3: Combine boxes of $Q^{[j]},\ j = 1, \ldots, p$ having the same common variable values to get the list Q of full dimensional final boxes. The full dimensional RUpT and monotonicity tests are applied.

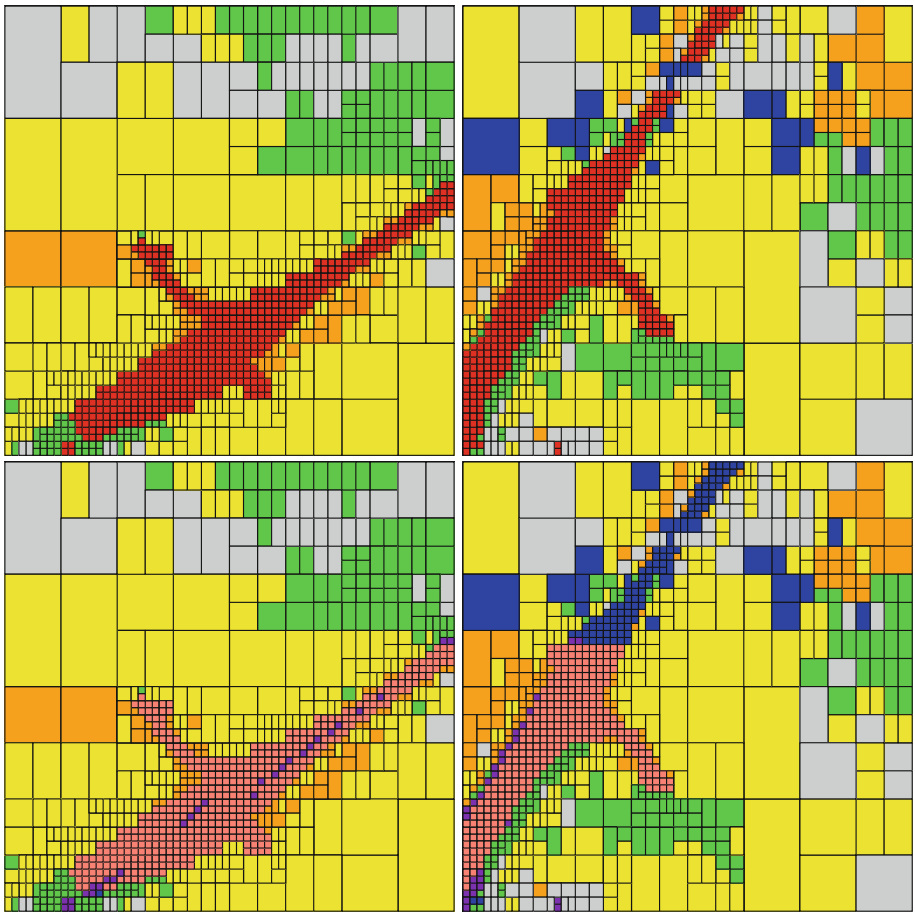

Fig. 1 GP3 boxes in phase 1 (*upper row*) and phase 2 (*bottom row*) of the IBB–DSP algorithm with $\epsilon = 10^{-1}$. $f^{[1]}(x_1, x_3)$ is at *left hand side* and $f^{[2]}(x_2, x_3)$ is at *right hand side*. *Colors of boxes* for the successful deletion: RUpT [RangeUp-test (*gray*) and Cutoff-test (*green*)], SubRUpT (*yellow*), NCVMT (*orange*), UVCV (*blue*), phase 1 final boxes (*red*), phase 2 RangeUp-test (*violet*), phase 2 final boxes (*salmon*). (Color figure online)

7 Numerical illustration

A set of 13 test functions with a common variable has been composed. The Appendix describes the characteristics of these functions. The first five test problems are three-dimensional. This facilitates obtaining a graphical output like Fig. 1. Example 1 is a simple test function created specifically for this work. Most of the problems have been constructed combining two well known global optimisation test functions by sharing one variable. For instance, problem Nr. 3 (L5P) is the combination of Levy-5 and Price functions, and problem Nr. 6 (G5G5) is the sum of two Griewank-5 functions. The algorithm was run using two different values for accuracy ϵ in the termination criterion, $\epsilon = 10^{-3}$ and $\epsilon = 10^{-6}$.

The algorithms were run using version 2.4.0 of C-XSC for interval arithmetic and interval automatic differentiation [2]. In order to limit the negative effects of the clustering problems

Table 1 Results of IBB and IBB–DSP algorithms for the Nr. 2 (GP3) test function

	$\epsilon = 10^{-3}$	$\epsilon = 10^{-6}$
IBB algorithm		
RUpT	40,777	44,615
Monot. Test	5,398	6,997
$\|Q\|$	141	140
$[lb, f^U]$	[64.994799, 65.000023]	[64.999999, 65.000001]
CPU	6.80	7.61
Effort	416,836	465,760
IBB–DSP algorithm phase 1		
RUpT	2,628	3,683
SubRUpT	886	1,334
UVCV	232	1,013
NCVMT	962	2,703
$\|Q^{[1]}\|$	162	147
$\|Q^{[2]}\|$	121	129
$[lb^{[1]}, f^U]$	[57.864553, 65.000023]	[62.408512, 65.000001]
IBB–DSP algorithm phase 2		
RUpT	18	20
UVCV	102	100
$\|Q^{[1]}\|$	79	75
$\|Q^{[2]}\|$	84	81
$[lb^{[1]}, f^U]$	[64.996493, 65.000023]	[64.999999, 65.000001]
IBB–DSP algorithm phase 3		
RUpT	42	35
Monot.Test	0	0
$\|Q\|$	137	133
$[lb, f^U]$	[64.996493, 65.000002]	[64.999999, 65.000001]
CPU	2.08	3.67
Effort	35,823	63,894

[3], the isotone inclusion function $F_\cap$ is applied for the computation of bounds, see (5). We used an Intel Core Duo T2300 1.66 GHz processor with 1.5 Gb of RAM.

We first analyse one instance to get hold of advantages and disadvantages of running the complete dimensional IBB compared to the DSP–IBB algorithm. Table 1 shows the behaviour of the IBB and IBB–DSP algorithms on test function GP3. It shows the performance of the IBB algorithm above and IBB–DSP algorithm below divided into phases 1, 2 and 3. The following performance indicators are measured:

– $|Q|$: the number of final n-dimensional boxes.
– $[lb, f^U]$: final interval including f^*.
– CPU: execution time in seconds.
– The effort value was measured as follows:

- For IBB = $\#F_{\cap} + n \cdot \#F'_{NIE}$, where # stands for number of evaluations.
- For IBB–DSP= $\#F_{\cap} + n \cdot \#F'_{NIE} + \sum_{i=1}^{p} \#F_{\cap}^{[i]} + n^{[j]} \cdot \#F^{[i]\prime}_{NIE}$.
  Notice that the cost to evaluate $F_{\cap}$ is different to evaluate $F_{\cap}^{[i]}$.

Additional information in Table 1 shows the number of boxes rejected by the tests presented in Sects. 3 and 6.3. Notice that phase 3 of the IBB–DSP uses rejection tests of the IBB algorithm. The number of final boxes $|Q^{[i]}|$ and the interval including f^* are also shown for phases 1 and 2. Any sub-function lower bound $lb^{[i]}$ (see (6)) can be used to obtain the interval including f^* in phases 1 and 2.

The performance of the IBB–DSP algorithm, in terms of effort and CPU time, is better than the IBB algorithm for the GP3 test function. Besides, the IBB–DSP algorithm obtains a better final inclusion of f^* for $\epsilon = 10^{-3}$. We first focus on possible advantages of the decomposed perspective:

1. Decomposition leads to lower dimensional problems.
2. Separation of terms may lead to less occurrences of the same variable. This helps to diminish overestimations of the real function range.
3. The new rejection tests for sub-functions (SubRUpT, NCVMT and UVCV), perform well due to point 2.
4. In IBB–DSP, the evaluated point trying to update f^U is restricted in the common component, to $X^{[j]} \cap \Psi^{[k]}(X_{n^{[j]}}),\ j \neq k$.
5. RUpT in phase 1 of the IBB–DSP algorithm is not as effective as RUpT of the IBB algorithm, but it still rejects a large number of boxes.

Possible disadvantages that can be derived from the observations:

1. In phase 1, the interval enclosing f^* is larger than the final enclosing interval in phase 3. The lower bound of $f(EX^{[j]})$ obtained by $\underline{F}_{DFE}(EX^{[j]})$ in (11) does not only depend on $F^{[j]}(X^{[j]})$ but also in how many subboxes in $L^{[k]} \cup Q^{[k]},\ k \neq j$ have points in common with $X^{[j]}_{n^{[j]}}$. These subboxes determine $\Psi^{[k]}(X^{[j]}_{n^{[j]}})$, see (7). Fewer subboxes in $L^{[k]} \cup Q^{[k]}$ give better values of $\underline{F}_{DFE}(EX^{[j]})$. Therefore, in phase 3 the RUpT is effective in removing more boxes (Table 1).
2. Testing monotonicity in the common variable is left for phase 3.
3. Better values of $\underline{F}_{DFE}(EX^{[j]})$ are obtained when $\Psi^{[k]}(X^{[j]}_{n^{[j]}}) \in Q^{[k]}$ (final set) as compared to $\Psi^{[k]}(X^{[j]}_{n^{[j]}}) \in L^{[k]}$. Phase 2 improves some $\underline{F}_{DFE}(EX^{[j]})$ values, because $\Psi^{[k]}(X^{[j]}_{n^{[j]}}) \in Q^{[k]}$. RUpT is most effective at the end of the run in phase 2.
4. In phase 2 working with final boxes, the UVCV test is effective due to the following reasons:
   - In phase 1, one sub-function number j may result in final subboxes in $Q^{[j]}$, whereas subboxes of another sub-function number k that shares common variable values are stored in $L^{[k]}$. Not all of these subboxes have offspring that reach $Q^{[k]}$.
   - Rejection of boxes of one sub-function by RUpT in phase 2 (see point 2) allows the UVCV test to remove boxes for other sub-functions.

Figure 1 shows rejected and final subboxes in phases 1 and 2 for the IBB–DSP algorithm on the GP3 problem with $\epsilon = 10^{-1}$. It illustrates the points mentioned above.

In Tables 2 and 3, the effectiveness of the IBB and IBB–DSP algorithms is measured in terms of the interval containing f^* and the number of boxes in the final list $|Q|$ for two values of the accuracy. For both values of accuracy ϵ, the IBB–DSP algorithm obtains a smaller

Table 2 Effectiveness of the algorithms for $\epsilon = 10^{-3}$

Nr.	n_1	n_2	IBB		IBB–DSP	
			$[lb, f^U]$	$\lvert Q\rvert$	$[lb, f^U]$	$\lvert Q\rvert$
1	2	2	$[-85.000001, -84.990234]$	4	$[-85.000001, -84.995117]$	4
2	2	2	$[64.994799, 65.000023]$	141	$[64.996493, 65.000002]$	137
3	2	2	$[-172.2774, -172.2768]$	3	$[-172.2774, -172.2768]$	3
4	2	2	$[-123.743285, -123.743157]$	2	$[-123.743285, -123.743159]$	2
5	2	2	$[-168.6570, -168.6566]$	24	$[-168.6570, -168.6566]$	24
6	5	5	$[-0.00001, 1.199041 \cdot 10^{-14}]$	512	$[-0.00001, 1.332268 \cdot 10^{-14}]$	512
7	6	3	$[-9.652299, -9.629085]$	4	$[-9.652299, -9.629121]$	4
8	5	3	$[-35.954858, -35.954755]$	9	$[-35.954858, -35.954780]$	9
9	5	2	$[0.153744, 0.154892]$	1,366	$[0.153744, 0.154892]$	1,258
10	4	4	$[-20.939446, -20.939333]$	1	$[-20.939446, -20.939340]$	1
11a	3	2	$[-0.000171, 7.578985 \cdot 10^{-6}]$	1,101	$[-0.000129, 8.643538 \cdot 10^{-7}]$	569
11b	2	3	$[-0.000171, 7.578985 \cdot 10^{-6}]$	1,101	$[-6.879867 \cdot 10^{-5}, 8.643538 \cdot 10^{-7}]$	67
12	3	3	$[-118.021834, -118.020600]$	2	$[-118.021834, -118.020827]$	2
13	3	3	$[-29.000001, -28.996337]$	16	$[-29.000001, -28.998168]$	16

Table 3 Effectiveness of the algorithms for $\epsilon = 10^{-6}$

Nr.	n_1	n_2	IBB		IBB–DSP	
			$[lb, f^U]$	$\lvert Q\rvert$	$[lb, f^U]$	$\lvert Q\rvert$
1	2	2	$[-85.000001, -84.999990]$	4	$[-85.000001, -84.999995]$	4
2	2	2	$[64.999999, 65.000001]$	137	$[64.999999, 65.000001]$	133
3	2	2	$[-172.2770, -172.2769]$	3	$[-172.2770, -172.2769]$	3
4	2	2	$[-123.743164, -123.743163]$	2	$[-123.743164, -123.743163]$	2
5	2	2	$[-168.6567, -168.6566]$	24	$[-168.6567, -168.6566]$	24
6	5	5	$[-0.0000001, 1.199041 \cdot 10^{-14}]$	512	$[-0.0000001, 1.332268 \cdot 10^{-14}]$	512
7	6	3	$[-9.652141, -9.652111]$	16	$[-9.652141, -9.652118]$	16
8	3	3	$[-35.954789, -35.954788]$	9	$[-35.954789, -35.954788]$	9
9	5	2	$[0.154870, 0.154871]$	1,628	$[0.154870, 0.154871]$	1,628
10	4	4	$[-20.939350, -20.939349]$	2	$[-20.939350, -20.939349]$	2
11a	3	2	$[-1.630625 \cdot 10^{-10}, 7.227997 \cdot 10^{-12}]$	1,107	$[-1.223355 \cdot 10^{-10}, 8.242295 \cdot 10^{-13}]$	581
11b	2	3	$[-1.630625 \cdot 10^{-10}, 7.227997 \cdot 10^{-12}]$	1,107	$[-6.561152 \cdot 10^{-11}, 8.242295 \cdot 10^{-13}]$	67
12	3	3	$[-118.020864, -118.020863]$	2	$[-118.020864, -118.020863]$	2
13	3	3	$[-29.000001, -28.999996]$	1	$[-29.000001, -28.999998]$	1

or equal number of final boxes. Moreover, the IBB–DSP algorithm results into a smaller or equal range for f^*.

In Tables 4 and 5, the efficiency is measured in terms of effort, CPU time (in seconds) and maximum number of elements in the lists reached during algorithm execution. For an accuracy of $\epsilon = 10^{-3}$, the IBB–DSP algorithm requires less effort than the IBB algorithm for all problems and spends similar or less time for all problems apart from case Nr. 9. For an accuracy of $\epsilon = 10^{-6}$, IBB–DSP requires less effort than IBB for 6 out of 13 problems and consumes less time for 4 problems. For most of the problems, using higher accuracy ϵ results into an increase in the size of $L^{[j]}$ and $Q^{[j]}$. This makes it harder for the tests in the decomposed algorithm IBB–DSP to be effective, as was mentioned before.

The maximum number of the elements in the working and final lists affects directly the IBB–DSP efficiency with respect to CPU time. This can be observed from the case of applying an higher accuracies. As mentioned in disadvantage number 1 in page 1113, a larger number of elements in the working and final lists leads to a worse estimation of the lower bound. Additionally, far more computational effort is needed to find the $\Psi^{[j]}(Y)$ value defined in (7). One has to go through the complete list of the other subproblem to do so. For an accuracy of $\epsilon = 10^{-3}$, the maximum number of elements in the working and final lists is in general smaller when using IBB–DSP compared to IBB.

What can be observed is that for the high dimensional case Nr. 6 (with 9 variables), the advantage of using a decomposed approach is larger than for lower dimensional cases. Having 3 variables means here to have two subproblems in two dimensions, whereas the 9-dimensional case has two subproblems in 5-dimensional space. On the other hand, also the worst advantage of using decomposition is found in a higher dimension with function Nr. 7. This case is characterised by an imbalance in number of variables for the sub-functions. This causes an imbalance in the length of the lists for the sub-functions. Better branch-and-bound selection strategies should be developed to overcome the imbalance.

We remark from Tables 2, 3, 4, and 5 that two different decompositions of the same problem (case Nr. 11) yield different numerical results (case Nr. 11b gives better efficiency). This shows that the way the problem is decomposed affects the efficiency, depending on the resulting maximum working and final lists size.

8 Conclusions and future work

Interval B&B algorithms aim at guaranteed solutions for global optimisation problems. Such methods suffer from the curse of dimensionality. We had a look here at functions that are separable, but where the resulting sub-functions have variables in common. Can one use the underlying structure to obtain more efficient algorithms based on interval Branch-and-Bound? To answer this question, we derived properties that can be used in interval branch-and-bound algorithms. Good bookkeeping of (sorted) underlying lists gives the opportunity to derive as well local lower bounds (for each sub-function) as global lower bounds. It has been shown, that the monotonicity test for the individual (non-common) variables can easily be extended to this framework. To apply also a monotonicity test for the common variable over all sub-functions, requires further investigation in possible implementations.

The developed properties have been included in a new branch-and-bound algorithm as part of the selection, bounding and elimination rules.

Experimental results show a potential improvement in effort and CPU time that depends mainly on the number of stored subproblems during the algorithm execution. This shows possible advantages using a decomposed strategy over a full-dimensional strategy.

Table 4 Efficiency of the algorithms for $\epsilon = 10^{-3}$

Nr.	IBB				IBB–DSP						Gain	
	Effort	CPU	$\|L\|^U$	$\|Q\|^U$	Effort	CPU	$\|L^{[1]}\|^U$	$\|Q^{[1]}\|^U$	$\|L^{[2]}\|^U$	$\|Q^{[2]}\|^U$	Effort	CPU
1	1,972	0.02	35	4	1,273	0.02	14	4	27	10	*	*
2	416,836	6.80	6,411	141	35,823	2.08	709	162	479	121	*	*
3	5,293	0.13	65	3	3,346	0.08	68	12	64	13	*	*
4	17,389	0.42	368	2	5,078	0.14	86	39	88	32	*	*
5	51,814	1.25	1,038	24	9,737	0.26	163	72	110	102	*	*
6	1,946,092	33.88	512	512	89,643	1.50	32	32	32	32	*	*
7	14,954	0.32	50	4	12,913	0.28	28	58	86	48	*	*
8	16,386,891	514.44	251,654	9	608,927	426.95	178	10,936	11,248	36	*	*
9	270,676	2.85	1,246	1,366	148,902	4.98	887	979	101	1,044	*	
10	3,299	0.14	83	1	1,946	0.06	19	3	18	5	*	*
11a	357,149	2.46	2,244	1,101	65,286	2.43	382	409	119	879	*	*
11b	357,149	2.46	2,244	1,101	6,920	0.09	43	62	29	9	*	*
12	54,952	0.50	568	2	7,020	0.11	94	43	92	16	*	*
13	9,946	0.07	33	16	2,572	0.03	17	5	12	5	*	*

∗The gain of the IBB–DSP algorithm

Table 5 Efficiency of the algorithms for $\epsilon = 10^{-6}$

Nr.	IBB				IBB–DSP						Gain	
	Effort	CPU	$\|L\|^U$	$\|Q\|^U$	Effort	CPU	$\|L^{[1]}\|^U$	$\|Q^{[1]}\|^U$	$\|L^{[2]}\|^U$	$\|Q^{[2]}\|^U$	Effort	CPU
1	3,187	0.02	55	4	2,064	0.03	14	4	27	13	*	
2	465,760	7.61	6,411	140	63,894	3.67	709	147	479	129	*	*
3	6,274	0.16	65	3	27,137	1.02	96	328	356	378		
4	17,866	0.44	368	2	52,360	3.83	392	795	620	660		
5	58,132	1.40	1,038	24	171,670	14.42	2,232	2,412	778	1,931		
6	2,913,772	50.68	512	512	131,243	2.15	32	32	32	32	*	*
7	30,439	0.64	59	16	594,826	92.76	968	2,147	2,762	6,644		
8	16,392,624	515.60	251,654	9	682,367	444.22	570	11,070	11,246	810	*	*
9	1,623,166	17.0	3,720	1,628	5,665,430	7329.00	26,475	28,032	2,007	23,882		
10	4,523	0.19	83	2	6,599	0.21	19	23	18	26		
11a	821,954	5.68	2,298	1,107	148,783	12.41	382	414	400	2154	*	
11b	821,954	5.68	2,298	1,107	10,455	0.13	42	62	48	9	*	*
12	56,252	0.51	568	2	72,593	3.40	446	553	443	521		
13	16,901	0.12	41	1	4,354	0.05	17	5	12	5	*	*

∗The gain of the IBB–DSP algorithm

Experiments have been designed in a controlled way, varying test instances that can be composed in an artificial way from existing test cases. As a future work, it is interesting to apply the derived theory to real problems where more than one common variable can appear. Additionally, this paper facilitates the development of new variants and strategies to increase efficiency and their application to solve design problems such as in [9].

Appendix : Description of the problems

The following functions are combined as components:

$$G2(x_1, x_2) = \frac{x_1^2 + x_2^2}{200} - \cos(x_1) \cdot \cos\left(\frac{x_2}{\sqrt{2}}\right) + 1$$

$$G5(x_1, x_2, x_3, x_4, x_5) = \sum_{i=1}^{5} \frac{x_i^2}{4000} - \prod_{i=1}^{5} \cos\left(\frac{x_i}{\sqrt{2}}\right) + 1$$

$$GP(x_1, x_2) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$$
$$\times[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$$

$$HM3(x_1, x_2) = -\sum_{i=1}^{2}\sum_{j=1}^{5} j \sin((j+1)x_i + j)$$

$$HM4(x_1, x_2, x_3) = -\sum_{i=1}^{3}\sum_{j=1}^{5} j \sin((j+1)x_i + j)$$

$$HP(x_1, x_2, x_3, x_4, x_5, x_6) = (((x_1 - x_4)^2 + (x_2 - x_5)^2 + (x_3 - x_6)^2)^{-3} - 0.5)^2$$

$$Levy3(x_1, x_2) = \sum_{i=1}^{5} i \cos((i+1)x_1 + i) \sum_{j=1}^{5} j \cos((j+1)x_2 + j)$$

$$Levy5(x_1, x_2) = \sum_{i=1}^{5} i \cos((i+1)x_1 + i) \sum_{j=1}^{5} j \cos((j+1)x_2 + j)$$
$$+ (x_1 + 1.42513)^2 + (x_2 + 0.80032)^2$$

$$Price(x_1, x_2) = (2x_1^3x_2 - x_2^3)^2 + (6x_1 - x_2^2 + x_2)^2$$

$$RB(x_1.x_2) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2$$

$$SHCB(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$

$$Sk(x_1, x_2, x_3, x_4) = -\sum_{i=1}^{k} \frac{1}{(x - a_i)(x - a_i)^T + c_i}, k = 5, 7, 10$$

with coefficients:

i	a_i	c_i
1	(4.0,4.0,4.0,4.0)	0.1
2	(1.0,1.0,1.0,1.0)	0.2
3	(8.0,8.0,8.0,8.0)	0.2
4	(6.0,6.0,6.0,6.0)	0.4
5	(3.0,7.0,3.0,7.0)	0.4
6	(2.0,9.0,2.0,9.0)	0.6
7	(5.0,5.0,3.0,3.0)	0.3
8	(8.0,1.0,8.0,1.0)	0.7
9	(6.0,2.0,6.0,2.0)	0.5
10	(7.0,3.6,7.0,3.6)	0.5

The generated test cases are:

1. Example1. Dimension: 3. Search space: $[-10, 10]^3$.
   $f^{[1]} = x_1^2 + x_1x_3 + \frac{1}{2}x_3^2 + x_3$, $f^{[2]} = x_2^2 - 2x_2x_3$
   $x^* = (5.0, -10.0, -10.0)$
   $f^* = -85.0$
2. GP3. Dimension: 3. Search space: $[-2, 2]^3$.
   $f^{[1]} = \mathrm{GP}(x_1, x_3)$, $f^{[2]} = \mathrm{GP}(x_2, x_3)$
   $x^* \in \{(-0.6, -0.6, -0.4), (-0.4, -0.4, -0.6)\}$
   $f^* = 65.0$
3. L5P. Dimension: 3. Search space: $[-10, 10]^3$.
   $f^{[1]} = \mathrm{Levy5}(x_1, x_3)$, $f^{[2]} = \mathrm{Price}(x_2, x_3)$
   $x^* = (-1.306853, 0.793737, -1.423764)$
   $f^* = -172.276922$
4. L5Pr. Dimension: 3. Search space: $[-10, 10]^3$.
   $f^{[1]} = \mathrm{Levy5}(x_1, x_3)$, $f^{[2]} = \mathrm{Price}(x_3, x_2)$
   $x^* = (-0.352130, 0.419827, -0.784273)$
   $f^* = -123.743163$
5. SHCBL3. Dimension: 3. Search space: $[-10, 10]^3$.
   $f^{[1]} = \mathrm{SHCB}(x_1, x_3)$, $f^{[2]} = \mathrm{Levy3}(x_3, x_2)$
   $x^* \in \{(1.733479, -7.589893, -1.417771),(1.733479, -1.306707, -1.417771), (1.733479, 4.976477, -1.417771)\}$
   $f^* = -168.656637$
6. G5G5. Dimension: 9. Search space: $[-600, 600]^9$.
   $f^{[1]} = \mathrm{G5}(x_1, \ldots, x_4, x_9)$, $f^{[2]} = \mathrm{G5}(x_5, \ldots, x_9)$
   $x^* = (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)$
   $f^* = 0.0$
7. HPHM4. Dimension: 8. Search space: $[0.8, 1.2] \times [0.4, 0.8]^2 \times [0.4, 0.6]^5$.
   $f^{[1]} = \mathrm{HP}(x_1, \ldots, x_5, x_8)$, $f^{[2]} = \mathrm{HM4}(x_6, x_7, x_8)$
   $x^* = (1.2, 0.8, 0.8, 0.4, 0.57, 0.57, 0.54)$
   $f^* = -9.652140$
8. L8HM4. Dimension: 5. Search space: $[-10, 10]^5$.
   $f^{[1]} = \mathrm{Levy8}(x_1, x_2, x_5)$, $f^{[2]} = \mathrm{HM4}(x_3, x_4, x_5)$
   $x^* \in \{ (1, 1, 5.79, 5.79, -0.49), (1, 1, 5.79, -6.77, -0.49), (1, 1, -6.77, 5.79, -0.49),$

(1, 1, −6.77, −6.77, −0.49), (1, 1, 5.79, −0.49, −0.49), (1, 1, −0.49, 5.79, −0.49), (1, 1, −0.49, −6.77, −0.49), (1, 1, −6.77, −0.49, −0.49), (1, 1, −0.49, −0.49, −0.49)}
$f^* = -35.95478$

9. RB5G2. Dimension: 6. Search space: $[-2, 2]^6$.
$f^{[1]} = \text{RBx5}(x_1, \ldots, x_4, x_6)$, $f^{[2]} = \text{G2}(x_5, x_6)$
$x^* = (0.97, 0.95, 0.9, 0.81, 0.0, 0.66)$
$f^* = 0.15487$
10. S10S7. Dimension: 7. Search space: $[0, 10]^7$.
$f^{[1]} = \text{S10}(x_1, x_2, x_3, x_7)$, $f^{[2]} = \text{S7}(x_4, x_5, x_6, x_7)$
$x^* = (4,4,4,4,4,4,4)$
$f^* = -20.939349$
11. Colville. Dimension: 4. Search space:$[-10, 10]^4$.
$fa^{[1]} = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + 19.8(x_2 - 1)(x_4 - 1) + 10.1((x_2 - 1)^2 + (x_4 - 1)^2)$
$fa^{[2]} = (x_3 - 1)^2 + 90(x_3^2 - x_4)^2$
$fb^{[1]} = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + 10.1(x_2 - 1)^2$
$fb^{[2]} = (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1(x_4 - 1)^2 + 19.8(x_2 - 1)(x_4 - 1)$
$x^* = (1,1,1,1)$
$f^* = 0.0$
12. Example 2. Dimension: 5. Search space: $[-5, 5]^5$.
$f^{[1]} = x_5 \sum_{i=1}^{2} x_i \sin x_i^2$
$f^{[2]} = x_5 \sum_{i=3}^{5} x_i \sin x_i^2$
$x^* \in \{(-4.8562, -4.8562, -4.8562, -4.8562, -4.8570), (4.8562, 4.8562, 4.8562, 4.8562, 4.8570)\}$
$f^* = -118.02086$
13. Example3. Dimension: 5. Search space: $[-5, 5]^5$.
$f^{[1]} = \sum_{i=1}^{2}(x_i - 1)^2 - \sum_{i=1}^{2} x_5 x_i$
$f^{[2]} = \sum_{i=3}^{5}(x_i - 1)^2 - \sum_{i=1}^{2} x_5 x_i$
$x^* = (3.5, 3.5, 3.5, 3.5, 5.0)$
$f^* = 0.0$

References

1. Baumann, E.: Optimal centered forms. BIT **28**(1), 80–87 (1988)
2. C-XSC: University Wuppertal. http://www.math.uni-wuppertal.de/~xsc/.
3. Du, K., Kearfott, R.B.: The cluster problem in multivariate global optimization. J. Glob. Optim. **5**(3), 253–265 (1994)
4. Hansen, E.R., Walster, G.W.: Global Optimization Using Interval Analysis. Marcel Dekker, New York (2004)
5. Hansen, P., Lagouanelle, J.-L., Messine, F.: Comparison between Baumann and admissible simplex forms in interval analysis. J. Glob. Optim. **37**(2), 215–228 (2007)
6. Kearfott , R.B.: Rigorous Global Search: Continuous Problems. Kluwer, Dordrecht (1996)
7. Markot, M.C., Fernandez, J., Casado, L.G., Csendes, T.: New interval methods for constrained global optimization. Math. Program. Ser. A **106**(2), 287–318 (2006)
8. Messine, F., Lagouanelle, J.-L.: Enclosure methods for multivariate differentiable functions and application to global optimization. J. Univers. Comput. Sci. **4**(6), 589–603 (1998)
9. Messine, F., Nogarede, B.: Optimal design of multi-airgap electrical machines: an unknown size mixed-constrained global optimization formulation. IEEE Trans. Magn. **42**(12), 3847–3853 (2006)
10. Moore, R.E.: Interval Analysis. Prentice-Hall, New Jersey (1966)
11. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to Interval analysis. SIAM, Philadelphia (2009)
12. Neumaier, A., Shcherbina, O., Huyer, W., Vinko, T.: A comparison of complete global optimization solvers. Math. Program. Ser. B **103**(2), 335–356 (2005)

13. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (1999)
14. Ratschek, H., Rokne, J.: Computer Methods for the Range of Functions. Ellis Horwood, Chichester (1984)
15. Ratschek, H., Rokne, J.: New Computer Methods for Global Optimization. Ellis Horwood, Chichester (1988)
16. Tóth, B., Casado, L.G.: Multi-dimensional pruning from Baumann point in an interval global optimization algorithm. J. Glob. Optim. **38**(2), 215–236 (2007)
17. Tóth, B., Csendes, T.: Empirical investigation of the convergence speed of inclusion functions in a global optimization context. Reliab. Comput. **11**(4), 253–273 (2005)