Interval Tests and Contractors Based on Optimality Conditions for Bound-Constrained Global Optimization

Laurent Granvilliers

Interval Tests and Contractors Based on Optimality Conditions for Bound-Constrained Global Optimization

Laurent Granvilliers

International Journal on Artificial Intelligence Tools c

Interval tests and contractors based on optimality conditions for bound-constrained global optimization

Laurent Granvilliers
*LS2N, Université de Nantes, France**
laurent.granvilliers@univ-nantes.fr

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

We study the problem of finding the global optimum of a nonlinear real function over an interval box by means of complete search techniques, namely interval branch-and-bound algorithms. Such an algorithm typically generates a tree of boxes from the initial box by alternating branching steps and contraction steps in order to remove non optimal sub-boxes. In this paper, we introduce a new contraction method that is designed to handle the boundary of the initial box where a minimizer may not be a stationary point. This method exploits the first-order optimality conditions and we show that it subsumes the classical monotonicity test based on interval arithmetic. A new branch-and-bound algorithm has been implemented in the interval solver Realpaver. An extensive experimental study based on a set of standard benchmarks is presented.

Keywords: Bound-constrained global optimization; Continuous optimization; Branch-and-bound algorithm; Interval methods; Monotonicity test; Constraint propagation

1. Introduction

Bound-constrained global optimization is the problem of finding the global optimum of a nonlinear real function over an interval box. This problem can be defined as

$$\boxed{\begin{array}{ll}\text{minimize} & f(x)\\ \text{subject to} & l \leq x \leq u\end{array}} \tag{1}$$

where $x = (x_1, \ldots, x_n)$ is a vector of real variables,

$$\Omega = [l_1, u_1] \times \cdots \times [l_n, u_n] \subseteq \mathbb{R}^n \tag{2}$$

is an interval box and $f : \Omega \to \mathbb{R}$ is a nonlinear differentiable real function. Every inequality $l_i \leq x_i$ or $x_i \leq u_i$ is a bound constraint. A global minimizer is a point $x^* \in \Omega$ such that $f(x^*) \leq f(x)$ for all $x \in \Omega$. The value $f^* = f(x^*)$ is the global

*LS2N, Faculté des sciences et techniques, 2 chemin de la Houssinière, BP 92208, 44322 Nantes Cedex 3, France

minimum of f over Ω. Global optimization is a hard task in general since the function may be non-convex with many local optima.

The bound-constrained global optimization problem can be rigorously solved by interval branch-and-bound algorithms that are designed to calculate an enclosure $[L, U]$ of the global optimum at a given tolerance. Such an algorithm recursively splits and reduces the initial box Ω and every sub-box is processed by various interval techniques. A common approach consists of solving the system of equations $\nabla f(x) = 0$ that must be verified by every stationary point of f. The monotonicity test implemented in the pioneering Moore-Skelboe algorithm [22, 32] tries to prove that there is no stationary point in a given box. Moreover, it is possible to reduce a box with respect to these equations by means of the interval Newton method [13], constraint propagation [15] or a modified interval Newton method that solves a convex relaxation using linear programming techniques [19]. It is also common to refine the upper bound U of the global optimum by means of local optimization techniques [14]. Moreover, it is useful to consider another constraint $f(x) \leq U$ during the constraint propagation process [15]. Finally, efficient branching strategies may accelerate the overall convergence of the algorithm [8]. These techniques are reviewed in [23, 27].

Boundary boxes cannot be handled like interior boxes since a minimizer located on the boundary of Ω may not be stationary. It follows that the system of equations $\nabla f(x) = 0$ does not hold in general, and specific techniques have been designed. First of all, the monotonicity test can be used to fix the value of a variable if the function is proved to be monotone in some coordinate direction. However, even if it is monotone, this test may be weak due the inherent pessimism of interval computations. As a consequence, it may be hard to remove the boundary of Ω for large boxes during the early stages of the search process. More recently, Puranik and Sahinidis [26] proposed to calculate an enclosure of the set of stationary points in a given boundary box, i.e., to solve the system of equations $\nabla f(x) = 0$, and to check the initial bounds following a probing strategy. This efficient strategy has been implemented in the famous global solver BARON [33].

In this paper, we propose a new interval contraction method for processing boundary boxes in an interval branch-and-bound algorithm. As done in [26], the first step consists of solving the system of equations $\nabla f(x) = 0$. We propose here to adapt the solving technique to the problem difficulty and we have implemented different interval-based consistency techniques [4, 5, 18]. In the second step, an accurate interval-based monotonicity test favorably replaces the probing strategy. An implementation has been done in the interval solver Realpaver [12]. An experimental study shows that several difficult problems become solvable in reasonable time. This paper is an extension of work presented in [11]. In particular, we have implemented and tested new solving strategies and the set of benchmarks has been extended, including 4 times more problems and bigger problems with at most 10^4 variables. As a consequence, the experimental study is much stronger. Last but not least, the related work is more detailed.

The rest of this paper is organized as follows. Section 2 introduces the interval branch-and-bound framework. The main contribution is described in Section 3. The experimental results are presented in Section 4, followed by a conclusion.

2. Interval branch-and-contract algorithms

2.1. *Interval arithmetic*

An interval $X = [a, b]$ represents the set of real numbers $\{x : a \leq x \leq b\}$. An interval $[a, b]$ is empty if we have $a > b$. We define the following operations given a non empty interval.

$$\begin{array}{lll} i. & \operatorname{wid} X = (b-a) & \text{(width)} \\ ii. & \operatorname{rad} X = (b-a)/2 & \text{(radius)} \\ iii. & \operatorname{mid} X = (b+a)/2 & \text{(midpoint)} \end{array}$$

Let $\mathbb{I}$ denote the set of intervals. An interval box $X \in \mathbb{I}^n$ is a Cartesian product of intervals $X_1 \times \cdots \times X_n$. A box X is empty if at least one X_i is empty. We define the following operations given a non empty box.

$$\begin{array}{lll} i. & \operatorname{wid} X = \max\{\operatorname{wid} X_i : 1 \leq i \leq n\} & \text{(width)} \\ ii. & \operatorname{mid} X = (\operatorname{mid} X_1, \ldots, \operatorname{mid} X_n) & \text{(midpoint)} \end{array}$$

Given a set of real numbers $S \subseteq \mathbb{R}$, the interval hull of S is defined as the interval $\operatorname{hull} S = [\inf S, \sup S]$. Given a set $S \subseteq \mathbb{R}^n$, the interval hull of S is defined as the interval box $\operatorname{hull} S = \operatorname{hull} S_1 \times \cdots \times \operatorname{hull} S_n$, each S_i being the i-th projection of S.

The interval arithmetic operations are defined as follows. Let $op : D \subseteq \mathbb{R}^n \to \mathbb{R}$ be an operation over the real numbers. Given an interval box $X \in \mathbb{I}^n$, the corresponding interval operation must return the interval

$$op(X) = \operatorname{hull}\{op(x) : x \in X \cap D\}$$

which is the tightest enclosure of the image of X under the operation. The study of monotonicity properties, limits and extrema of the operations leads to the following computational definitions.

$$\begin{array}{ll} [a,b] + [c,d] & = [a+c, b+d] \\ [a,b] - [c,d] & = [a-d, b-c] \\ [a,b] \times [c,d] & = [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}] \\ [c,d]^{-1} & = \left[d^{-1}, c^{-1}\right] \text{ if } 0 \notin [c,d] \end{array}$$

The elementary functions are extended similarly. Now let $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ be a real function. An interval function $F : \mathbb{I}^n \to \mathbb{I}$ is an *interval extension of* f if we have

$$(\forall X \in \mathbb{I}^n)\ (\forall x \in X \cap D)\ f(x) \in F(X).$$

This property is known as the *fundamental theorem of interval arithmetic*. It follows that the interval $F(X)$ is an enclosure of the range of f over X. Three different techniques are presented thereafter. In order to illustrate them, let $f(x_1, x_2) = x_1^2 - x_1 x_2 - x_2$ be a real function and let $X = [-2, 0] \times [-4, 2]$ be a box.

The *natural extension of f* is derived by replacing every real operation in a given expression of f by the corresponding interval operation. The range of f over X is equal to $[-2, 6]$. The evaluation of the natural extension

$$F(X) = [-2, 0]^2 - [-2, 0]\,[-4, 2] - [-4, 2]$$

leads to a weak enclosure $[-10, 12]$. The pessimism here comes from the decorrelation of the multiple occurrences of the variables, which is known as *the dependency problem of interval arithmetic*.

The *mean value extension of f over X* derives from a Taylor approximation of f around a point $c \in X$ assuming that f is differentiable over X. Let G be an interval extension of the gradient of f. Then we have

$$(\forall x \in X)\ f(x) \in f(c) + G(X)^T \cdot (x - c).$$

In the previous example, given $c = \operatorname{mid} X$, a more precise enclosure $[-8, 10]$ results from the evaluation of the mean value extension

$$F_c(X) = 1 + ([-6, 4], [-1, 1]) \cdot \begin{pmatrix} [-1, 1] \\ [-3, 3] \end{pmatrix}.$$

Affine arithmetic leads to another kind of interval extension, as follows. The affine form of a variable x_i lying in X_i is defined as $\operatorname{mid} X_i + \operatorname{rad} X_i \times \varepsilon_i$ where $-1 \leq \varepsilon_i \leq 1$ is an affine variable associated with x_i. It follows the affine forms

$$\widehat{x_1} = -1 + \varepsilon_1 \quad \text{and} \quad \widehat{x_2} = -1 + 3\varepsilon_2.$$

Each variable ε_i catches the linear dependences between the multiple occurrences of x_i in f, hence counteracting the dependency problem of interval arithmetic. Every nonlinear term is linearized such that a new affine variable is introduced to bound the linearization error. The square of the affine form of x_1 is equal to $1 - 2\varepsilon_1 + \varepsilon_1^2$ and the nonlinear term ε_1^2 is replaced by $0.5 + 0.5\varepsilon_3$, which leads to

$$\widehat{x_1^2} = 1.5 - 2\varepsilon_1 + 0.5\varepsilon_3.$$

The product of the affine forms of x_1 and x_2 is equal to $1 - \varepsilon_1 - 3\varepsilon_2 + 3\varepsilon_1\varepsilon_2$ and the nonlinear term $3\varepsilon_1\varepsilon_2$ is replaced by $3\varepsilon_4$, which leads to

$$\widehat{x_1 x_2} = 1 - \varepsilon_1 - 3\varepsilon_2 + 3\varepsilon_4.$$

The affine form of f over X is eventually calculated as

$$\widehat{f} = \widehat{x_1^2} - \widehat{x_1 x_2} - \widehat{x_2} = 1.5 - \varepsilon_1 + 0.5\varepsilon_3 + 3\varepsilon_4.$$

Replacing each ε_i in $\widehat{f}$ by its domain permits to calculate the new enclosure

$$F_a(X) = 1.5 - [-1, 1] + 0.5\,[-1, 1] + 3\,[-1, 1] = [-3, 6].$$

In general, many interval extensions can be combined in order to calculate more precise enclosures. These techniques can be useful to derive lower bounds for the global optimization problem.

2.2. *Interval contractors*

The contractor programming framework [6] is an abstract formalism used to describe here the interval contraction techniques. Let $c(x)$ be a constraint over a vector of variables $x \in \mathbb{R}^n$. An operator Γ on boxes is an *interval contractor for* c if we have

$$(\forall X \in \mathbb{I}^n)\ \{x \in X : c(x)\} \subseteq \Gamma(X) \subseteq X. \tag{3}$$

The purpose of Γ is to eliminate facets of X that contain no solution of c. Moreover, if $\Gamma(X)$ is empty then it is proved that c has no solution in X.

Several classical techniques are presented thereafter. An *hc4 contractor for a constraint* c, also called HC4Revise [4], is a two-step algorithm traversing the tree-representation of c. The first phase is an interval evaluation from the leaves to the root. The second phase contracts the domains from the root to the leaves, as illustrated by Fig. 1. This operator is cheap and potentially very useful.

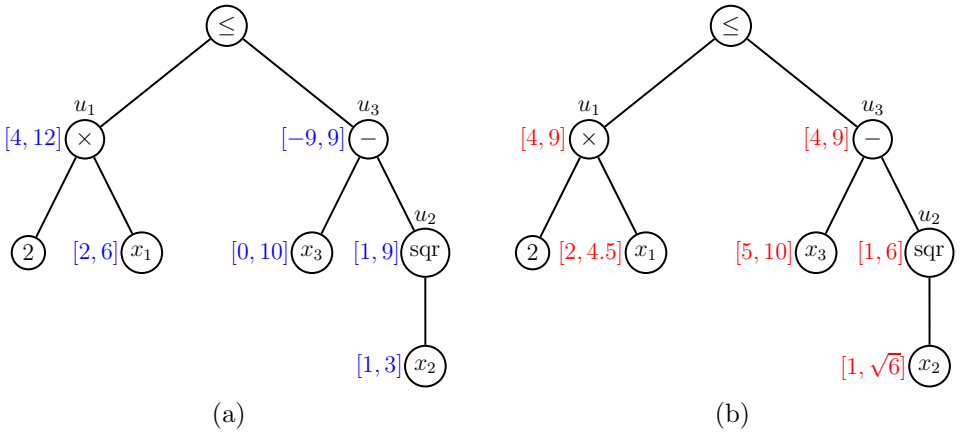

Fig. 1. An hc4 contractor applied to the constraint $2x_1 \leq x_3 - x_2^2$ and the box $[2,6]\times[1,3]\times[0,10]$. Fig. (a) illustrates the evaluation phase. Fig. (b) depicts the domains computed during the second phase. For instance, we have $u_1 \leq u3$ with $u_1 \in [4,12]$ and $u_3 \in [-9,9]$ at the root node. Hence it comes $u_1 \in [4,9]$ since u_1 cannot be greater than the maximum of u_3. Then the process follows in the left-hand sub-tree. We have $2x_1 = u_1$ with $x_1 \in [2,6]$ and $u_1 \in [4,9]$. By an inversion of this equation, it comes $x_1 = u_1/2$ and a new domain $x_1 \in [2,5]$ is easily derived. The right-hand sub-tree is processed in the same way. The reduced box $[2,4.5] \times \left[1,\sqrt{6}\right] \times [5,10]$ is obtained.

A *bc3 contractor*, also called BC3Revise [5], combines a bisection algorithm with the univariate interval Newton operator. Given a univariate equation $f(x) = 0$ and an interval domain X, a new interval $[a,b] \subseteq X$ is derived such that a is the leftmost zero of f in X and b is the rightmost zero of f in X. The bisection algorithm divides the initial interval in order to separate the zeros and the Newton operator is able under conditions to converge with a quadratic convergence rate towards a zero. For example, let $f(x) = x^3 - 4x^2 + x + 3 = 0$ and let $X = [-1,4]$. Fig. 2 shows the subdivision of X in 5 intervals and the application of two Newton steps

applied to X_1 and X_5 that find the outermost zeros. A Newton step consists of enclosing the curve of f by a cone given by its mean value extension. This cone is intersected with the current domain in order to derive a new domain. The interval returned by the bc3 contractor is approximatively equal to $[-0.69963, 3.4606]$ while an hc4 contractor applied to this problem returns the interval $[-1, 3.9579]$. The advantage of bc3 is to eliminate the dependency effect of interval arithmetic due to the multiple occurrences of x in f. However this operator is based on an iterative method embedded in a search process, hence being more expensive.

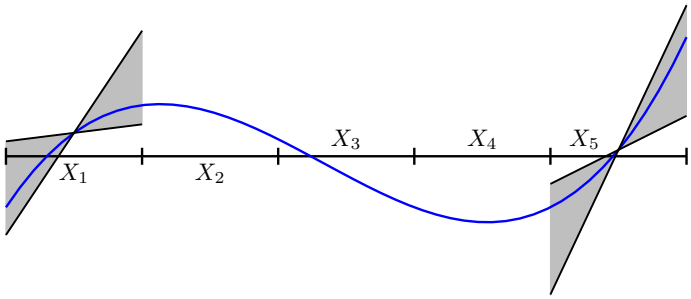

Fig. 2. Some steps of a bc3 contractor applied to the equation $x^3 - 4x^2 + x + 3 = 0$ and the interval $X = [-1, 4]$

Given an equation $f(x_1, \ldots, x_n) = 0$ and a box X, a bc3 contractor can be applied for each variable x_i to the univariate equation obtained from f by assigning each variable x_j but x_i to its domain X_j. Inequality constraints can also be processed by this technique.

A *bc4 contractor* [4] combines an hc4 contractor with bc3 contractors, as follows. Given a constraint and a box, the hc4 contractor is first applied to derive a new box. Then a bc3 contractor is applied for every variable occurring more than once in the constraint. The idea exploited here is that hc4 and bc3 contractors have the same contraction power for variables occurring only once in the constraint and hc4 is much cheaper than bc3. For constraints with only single occurrences of variables, bc4 is equivalent to hc4.

A *constraint propagation algorithm* aims at pruning a box with respect to a set of constraints. The main idea is to propagate the modifications of domains through the constraints until reaching a fixed-point. Algorithm 1 takes as input a set of contractors $\mathcal{S}$ and a box X. It returns a new box X such that $\Gamma(X) = X$ for every contractor $\Gamma \in \mathcal{S}$. The main idea is to maintain a set of active contractors $\mathcal{Q}$. Every contractor is inserted in $\mathcal{Q}$ at the beginning (line 2). Every active contractor is removed from $\mathcal{Q}$ before its application (lines 6 and 7). Then a non-active contractor is inserted in $\mathcal{Q}$ if it is associated with a constraint that depends on a variable x_i whose domain has been modified (line 11). This algorithm follows an AC3-like propagation strategy [20] and it can be improved to avoid slow convergences.

Algorithm 1: Constraint propagation algorithm

Input :
– set of contractors $\mathcal{S}$
– box $X = X_1 \times \cdots \times X_n$
Output: a new box included in X

```
let S_i = {Γ ∈ S | Γ depends on x_i} (i = 1, ..., n)
Q ← S                                          // set of active contractors
while Q is not empty and X is not empty do
    Y ← X                                      // save a copy of X
    do
        remove an element Γ from Q
        X ← Γ(X)                               // apply an active contractor to X
    while Q is not empty and X is not empty
    for i ← 1 to n do
        if X_i ≠ Y_i then
            insert every element of S_i in Q   // propagation step with respect to x_i
        end
    end
end
return X
```

From a theoretical point of view, constraint propagation can be explained in terms of chaotic iterations [1]. In particular, Algorithm 1 terminates in finite time if we consider intervals bounded by floating-point numbers since there are a finite number of boxes and every step is contracting. Finally, it is worth noticing that a propagation algorithm verifies the properties of an interval contractor.

Algorithm 1 is called hc4 or bc4 if it uses only hc4 or bc4 contractors. The hc4 algorithm can be extended to process constraint systems represented by directed acyclic graphs (DAG) also called computational graphs [31]. The contraction power is improved by intersecting the domain modifications at the shared operation nodes. For example, the system of equations $x^2 + y^2 = 2 \wedge y = x^2$ has two solutions $(-1, 1)$ and $(1, 1)$ in the box $[-10, 10]^2$. The hc4 algorithm leads to the new box $[-1.19, 1.19] \times [0.765, 1.42]$. This box is a weak enclosure of the solution set, which is due to the *locality effect* of propagation algorithms also identified as the *poor man's LP* strategy. In fact, constraints are considered one by one and not globally. Now, let $z = x^2$ represent the only shared operation node of this problem. Fig. 3 illustrates the first steps of the propagation process into the DAG. The final box $[-1, 1] \times \{1\}$ corresponds to the hull of the solution set, so it verifies the global consistency property. This technique reduces in general the locality effect, and this effect simply vanishes in this well-chosen example.

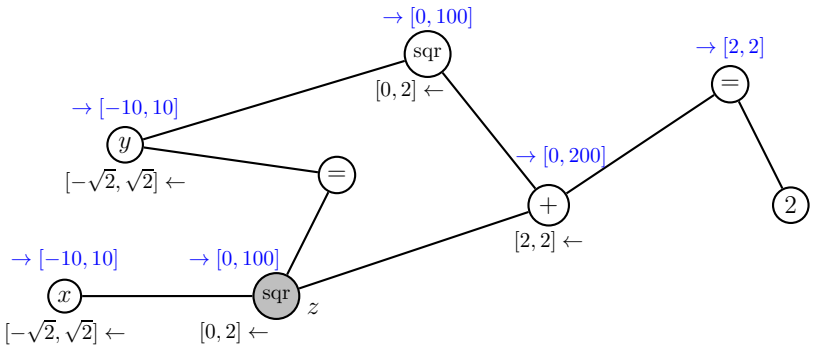

Fig. 3. First steps of the propagation process into the directed acyclic graph of the constraint system $x^2 + y^2 = 2 \wedge y = x^2$ given the box $[-10, 10]^2$. A right arrow $\rightarrow$ labels a domain calculated in the forward phase of hc4; a left arrow $\leftarrow$ labels a domain obtained in the backward phase.

2.3. *Branch-and-contract algorithm*

A *branch-and-contract algorithm* is a branch-and-bound algorithm taking as input a global minimization problem given by $f(x)$ and Ω and two tolerances $\epsilon_f > 0$ and $\epsilon_x > 0$. It returns an enclosure $[L, U]$ of the global minimum f^* such that we have $U \leq L + \epsilon_f$ and an enclosure of a global minimizer $x^* \in \Omega$ or the best feasible point found.

The general principle is to explore Ω in a deterministic way and to maintain a list of sub-boxes. Every box $X \subseteq \Omega$ is processed by several techniques described thereafter at an abstract level. Let F be an interval extension of f.

- A propagation algorithm $\Gamma : \mathbb{I}^n \rightarrow \mathbb{I}^n$ is associated with a set of constraints that must be verified by the global minimizers. Then X can be contracted as $\Gamma(X)$, hence removing non optimal parts of X or the whole box. Several categories of constraints can be used:
  - the upper bound constraint $f(x) \leq U$ since U must be an upper bound of the global minimum;
  - the gradient constraints
$$\frac{\partial f(x)}{\partial x_1} = 0, \ldots, \frac{\partial f(x)}{\partial x_n} = 0$$
for any X not on the boundary of Ω;
  - the convexity constraints
$$\frac{\partial f^2(x)}{\partial x_1^2} \geq 0, \ldots, \frac{\partial f^2(x)}{\partial x_n^2} \geq 0$$
for any X not on the boundary of Ω assuming that the function f is twice differentiable.
- The enclosure $[L, U]$ of the global optimum can be initialized by evaluating the interval extension F over Ω, since any global optimizer x^* must belong

to Ω and $f(x^*) \in F(\Omega)$ by property of an interval extension.

- The upper bound U can be refined as $\min(U, \max F(x))$ given any $x \in X$. It would be even better to apply a local optimization solver in order to derive good values of U during the early stages of the search process. It follows that the upper bound constraint is strengthened.
- A lower bound ℓ of f over X must be calculated. This can be done by evaluating an interval extension of f or by solving a convex relaxation of the optimization problem. Given the lower bound ℓ, the cut-off test leads to eliminate X if we have $\ell > U$ since the upper bound constraint $f(x) \leq U$ is violated for every $x \in X$.
- The selected box X at each step of the algorithm is such that the lower bound ℓ of f over X is the lowest one among the boxes to be processed (best-first strategy). Other strategies may use advantageously the upper bound U to diversify the search [24].
- A branch-and-contract algorithm follows a divide-and-conqueer strategy. Hence, X can be split into several sub-boxes if at least one of its components is an interval whose width is strictly greater than ϵ_x. There are several branching heuristics, for instance:
  - the simplest strategy just bisects the largest component of X;
  - a more efficient strategy in general consists of choosing the variable x_i with the maximum *smear value* $\operatorname{wid} X_i \times |G_i(X)|$ where G_i is an interval extension of the partial derivative of f with respect to x_i [8].

We see that many strategies can be derived from the general scheme depending on the choice of the different components. From a theoretical point of view, the convergence properties of interval branch-and-bound algorithms are discussed in [7]. As an example, we consider the so-called Shubert function defined by

$$f(x_1, x_2) = \left(\sum_{i=1}^{5} i \cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i \cos((i+1)x_2 + i)\right)$$

and the input domain is assigned to $-10 \leq x_1, x_2 \leq 10$. This function is depicted in Fig. 4. It is quite challenging for an interval-based algorithm for at least three reasons. There are many local and global minima, which can make difficult the proof of global optimality. The expression of f has many multiple occurrences of the variables. The expression of the gradient is as complex as the expression of f. It follows that the contraction techniques may be weak due to the dependency problem of interval arithmetic. Despite these drawbacks, it is possible to calculate an enclosure of the global minimum

$$f^* \in [-186.73091, -186.73090]$$

and an enclosure of one global minimizer

$$x^* \in [-7.0835065, -7.0835064] \times [-7.7083138, -7.7083137]$$

given the tolerances $\epsilon_f = 10^{-4}$ and $\epsilon_x = 10^{-8}$ after processing around 10^3 boxes in less than one second on a common laptop.

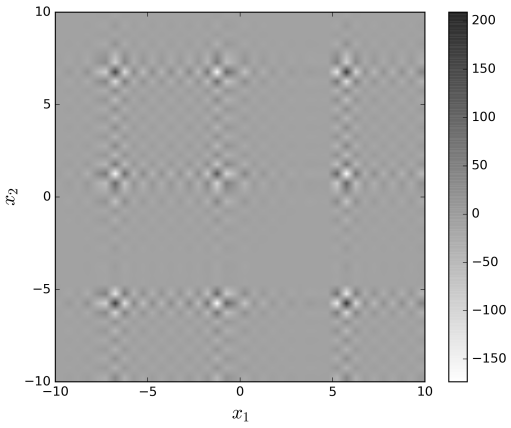

Fig. 4. Color map of the Shubert function.

3. Interval techniques based on optimality conditions

In the following, let $g = (g_1, \ldots, g_n)^T$ be the gradient of f. Let Γ_0 be a contractor associated with the upper bound constraint $f(x) \leq U$. Let Γ_i be a contractor associated with the equation $g_i(x) = 0$ for every i.

We say that a box X is an *interior box* if it is included in the interior of the initial box Ω. Otherwise, X is declared as a *boundary box*.

3.1. *Monotonicity test*

A continuous monotone function takes its extremal values over a domain at the domain bounds. Now, let X be a box and let i be a coordinate direction. The minimum of f over X is obtained at the left bound $x_i = \min X_i$ if it is increasing or it is obtained at the right bound $x_i = \max X_i$ if it is decreasing. In both cases, the value of x_i can be fixed.

The monotonicity property of f can be proved by means of interval arithmetic as follows. Let G_i be an interval extension of the partial derivative $g_i(x)$. Then, assuming that X and the domain of g_i overlap, f is increasing if we have

$$\min G_i(X) \geq 0 \tag{4}$$

or it is decreasing if we have

$$\max G_i(X) \leq 0. \tag{5}$$

The monotonicity test is particularly useful during the preprocessing step in order to reduce the problem dimension. For example, the *qrtquad* problem [9] has 120 variables. Let us consider the variable x_{12}, the derivative $g_{12}(x) = 8x_{12} + x_{120} - 120$ and the initial bounds $0 \leq x_{12}, x_{120} \leq 10$. Since the maximum of the interval

$$G_{12}([0,10],[0,10]) = 8\,[0,10] + [0,10] - 120 = [-120,-30] \tag{6}$$

is negative then the minimum of f is obtained at $x_{12} = 10$. In fact, 109 variables are fixed in this way. As a consequence, the search algorithm is applied to a simplified problem with only 11 variables.

3.2. *Local optimality conditions*

A global minimizer of the bound-constrained optimization problem must be a stationary point of the Lagrange function

$$L(x,\lambda,\mu) = f(x) + \sum_{i=1}^{n} \lambda_i (l_i - x_i) + \sum_{i=1}^{n} \mu_i (x_i - u_i) \tag{7}$$

where λ and μ are vectors of multipliers associated with the bound constraints. The Karush-Kuhn-Tucker conditions are defined as follows.

$$\text{KKT:} \begin{cases} g(x) - \lambda + \mu = 0 & \text{(stationarity)} \\ \lambda(l - x) = 0 & \text{(complementary slackness)} \\ \mu(x - u) = 0 & \text{(complementary slackness)} \\ l \leq x \leq u & \text{(primal feasibility)} \\ \lambda, \mu \geq 0 & \text{(dual feasibility)} \end{cases} \tag{8}$$

The stationarity condition states for each $i = 1, \ldots, n$ that

$$g_i(x) - \lambda_i + \mu_i = 0. \tag{9}$$

We distinguish three cases. First, when the bound constraints associated with x_i are inactive, the complementary slackness conditions imply that the multipliers λ_i and μ_i are equal to 0. Then it comes from Equation 9 that

$$g_i(x) = 0 \;\text{ if }\; l_i < x_i < u_i. \tag{10}$$

Second, when the bound constraint $x_i = l_i$ is active, it comes that $\mu_i = 0$ by complementary slackness since $x_i \neq u_i$ (assuming that $u_i > l_i$) and the dual feasibility condition states that $\lambda_i \geq 0$. Then it follows from Equation 9 that

$$g_i(x) \geq 0 \;\text{ if }\; x_i = l_i. \tag{11}$$

Third, by a similar argument, when the bound constraint $x_i = u_i$ is active, it comes that

$$g_i(x) \leq 0 \;\text{ if }\; x_i = u_i. \tag{12}$$

The optimality conditions 10–12 provide constraints that can be exploited by interval contraction techniques.

3.3. *Contraction techniques for boundary boxes*

The optimality condition (10) implies that any box X can be contracted as $\Gamma_i(X)$ if X_i is strictly included in Ω_i. It is thus possible to remove initial bounds for the other variables. Otherwise, the intervals X_i and Ω_i share at least one bound and the condition (10) is not valid anymore. We have the following results.

Lemma 3.1. *Let X be a box and let Y be the box $\Gamma_i(X)$ for some i. We have $g_i(x) \neq 0$ for all $x \in X \setminus Y$.*

Proof. *By property of Γ_i we have $\{x \in X : g_i(x) = 0\} \subseteq \Gamma_i(X)$. The result follows since we have $x \notin \Gamma_i(X)$ by assumption.*

Lemma 3.2. *Let X be a box such that $\min X_i = l_i$ for some i, let Y be the box $\Gamma_i(X)$ and let X^{i-} be the left facet $\{x \in X : x_i = l_i\}$. If $\min Y_i \neq l_i$ then we have $g_i(x) \neq 0$ for all $x \in X^{i-}$.*

Proof. *This result directly follows from Lemma 3.1 since X^{i-} does not intersect $\Gamma_i(X)$.*

Lemma 3.2 deals with the left bound of X_i. It proves that f is monotone over the facet X^{i-} under some conditions. A similar result can be stated for the right facet X^{i+} defined as $\{x \in X : x_i = u_i\}$ when $\max X_i = u_i$ and $\max Y_i \neq u_i$.

Lemma 3.3. *Let X be a box such that $\min X_i = l_i$ for some i and let Y be the box $\Gamma_i(X)$. If $\min Y_i \neq l_i$ and $g_i(x) < 0$ for all $x \in X^{i-}$ then no local minimizer belongs to the slice $\{x \in X : l_i \leq x_i < \min Y_i\}$.*

Proof. *It follows from Lemma 3.2 that f must be monotone over the facet X^{i-}. Assuming that f is strictly decreasing implies that the local optimality condition* (11) *is violated. This proves that there is no minimizer on X^{i-}. Moreover we have $g_i(x) \neq 0$ for all $x \in X$ such that $l_i < x_i < \min Y_i$ by property of Γ_i. It follows that the slice $\{x \in X : l_i < x_i < \min Y_i\}$ contains no minimizer since the local optimality condition* (10) *is violated. The proof is done.*

Lemma 3.3 deals with the left bound of X_i. It proves that this bound can be strictly contracted under some conditions. A similar result can be stated for the right bound when $\max X_i = u_i$, $\max Y_i \neq u_i$ and f is strictly increasing over X^{i+} by application of the optimality condition (12).

Enforcing Lemma 3.3 requires to determine the sign of the derivative $g_i(x)$ over X^{i-}. Since we know that g_i does not vanish over the facet then it suffices to consider any point $\hat{x} \in X^{i-}$. If we have

$$\max G_i(\hat{x}) < 0 \tag{13}$$

then f is proved to be strictly decreasing. This test is very accurate since the evaluation of G_i at a point $\hat{x}$ is not subject to the dependency problem of interval arithmetic. However, the result is in general a small interval due to the rounding

errors. In some singular cases when the slope of f is almost equal to 0, it may arise that the resulting interval contains 0. In such a case it could be interesting to consider different points of the facet in order to make a proof.

Theorem 3.1. Let X be a box, let i be a coordinate direction, and let Y be the box $\Gamma_i(X)$. Let $b_l \iff (\min X_i = l_i)$ and $b_r \iff (\max X_i = u_i)$ be two booleans. Let $d_l \iff b_l \wedge (\forall x \in X^{i-})\ g_i(x) < 0$ and $d_r \iff b_r \wedge (\forall x \in X^{i+})\ g_i(x) > 0$ be two booleans. The region in each of the following cases contains all the local minimizers of f within X.

i.	Y	if	$(\neg b_l)$	$\wedge$	$(\neg b_r)$						
ii.	Y	if	b_l	$\wedge$	$(\neg b_r)$	$\wedge$	d_l				
iii.	$Y \cup X^{i-}$	if	b_l	$\wedge$	$(\neg b_r)$	$\wedge$	$(\neg d_l)$				
iv.	Y	if	$(\neg b_l)$	$\wedge$	b_r			$\wedge$	d_r		
v.	$Y \cup X^{i+}$	if	$(\neg b_l)$	$\wedge$	b_r			$\wedge$	$(\neg d_r)$		
vi.	Y	if	b_l	$\wedge$	b_r	$\wedge$	d_l	$\wedge$	d_r		
vii.	$Y \cup X^{i-}$	if	b_l	$\wedge$	b_r	$\wedge$	$(\neg d_l)$	$\wedge$	d_r		
viii.	$Y \cup X^{i+}$	if	b_l	$\wedge$	b_r	$\wedge$	d_l	$\wedge$	$(\neg d_r)$		
ix.	$Y \cup X^{i-} \cup X^{i+}$	if	b_l	$\wedge$	b_r	$\wedge$	$(\neg d_l)$	$\wedge$	$(\neg d_r)$		

Proof. Y contains all the stationary points of f within X. The optimality condition (10) implies that Y must be part of the result, which is true for all cases. Moreover the facet X^{i-} must be part of the result if Lemma 3.3 is verified and it can be removed otherwise, which is true for all cases. The facet X^{i+} is handled similarly, which completes the proof.

Theorem 3.1 can be exploited in order to handle a boundary box in the branch-and-contract algorithm. Several variants can be defined depending on the following heuristics. First every contractor Γ_i can be implemented by different techniques. Second it is possible to split a union of boxes (cases *iii*, *v*, *vii*–*ix*) or to return their hull with the aim of limiting the number of boxes. Third, the facet X^{i-} (cases *iii*, *vii*, *ix*) may be contracted by considering one or both constraints $f(x) \leq U$ and $g_i(x) \geq 0$, or it may not be contracted (similar arguments for the facet X^{i+}).

Algorithm 2 implements a new contractor for boundary boxes. This contractor, denoted by $\partial\Gamma_i$, is parameterized by a contractor Γ_i associated with the equation $g_i(x) = 0$. With respect to the variants discussed above, we consider the hulls of unions of boxes (lines 15 and 18) in order to limit the number of boxes and the facets X^{i-} and X^{i+} are not contracted (lines 6, 8, 15 and 18). The correctness of Algorithm 2 is stated by the following theorem.

Theorem 3.2. Let $X \subseteq \Omega$ be a box and let $\partial\Gamma_i(X)$ be the box returned by Algorithm 2 given X as input. Then there is no point $x \in X \setminus \partial\Gamma_i(X)$ such that $f(x) = f^*$.

Proof. If $\Gamma_i(X)$ returns an empty box then f is proved to be monotone by property of Γ_i (line 4). In this case, the result derives from reliable interval-based monotonicity

tests based on the local optimality conditions. Otherwise, Y is assigned to the resulting box (line 13) and the facets X^{i-} and X^{i+} are added or not according to Theorem 3.1 (lines 15 and 18). This completes the proof.

Algorithm 2: Contractor $\partial\Gamma_i$ for boundary boxes

Input :
- objective function $f : \mathbb{R}^n \to \mathbb{R}$
- initial box $\Omega = [l_1, u_1] \times \cdots \times [l_n, u_n] \subseteq \mathbb{I}^n$
- boundary box $X \subseteq \Omega$
- contractor Γ_i associated with $g_i(x) = 0$

Output: a box $Z \subseteq X$ that contains all the global minimizers $x^* \in X$

```
1  b_l ← (min X_i = l_i)
2  b_r ← (max X_i = u_i)
3  Y ← Γ_i(X)                                   // contraction (stationarity constraint)
4  if Y = ∅ then                                // f monotone over X
5  |  if b_l and f increases on X then
6  |  |  Z ← X^{i-}                            // optimality condition (11)
7  |  else if b_r and f decreases on X then
8  |  |  Z ← X^{i+}                            // optimality condition (12)
9  |  else
10 |  |  Z ← ∅                                 // optimality condition (10)
11 |  end
12 else
13 |  Z ← Y                                    // all cases of Theorem 3.1
14 |  if b_l and min Y_i ≠ l_i and f increases on X^{i-} then
15 |  |  Z ← hull(Z ∪ X^{i-})                  // cases iii, vii, ix of Theorem 3.1
16 |  end
17 |  if b_r and max Y_i ≠ u_i and f decreases on X^{i-} then
18 |  |  Z ← hull(Z ∪ X^{i+})                  // cases v, viii, ix of Theorem 3.1
19 |  end
20 end
```

The contraction algorithm for boundary boxes can be implemented as a propagation algorithm managing the set of contractors $\{\Gamma_0, \partial\Gamma_1, \ldots, \partial\Gamma_n\}$. The correctness of this algorithm follows from the correctness of every contractor with respect the global optimization problem.

We will consider three minimization problems in order to illustrate the capabil-

ities of the new algorithm using hc4 contractors. Let

$$f(x_1, x_2, x_3) = (x_1 - 1)^2 + (x_2 - 1)^2 + (x_3 - 1)^2 - x_2x_1 - x_3x_2$$

be the 3-dimensional trid function and let $\Omega = [-9, 9]^3$. Let $U = 3$ be an upper bound of f^* obtained from an evaluation of f at the midpoint of Ω. First, Γ_0 has no effect. Second, $\partial\Gamma_1$ is able to contract the domain of x_1. It comes the new domain $[-3.5, 5.5]$. Moreover, the bound $x_1 = -9$ is discarded since a monotonicity test proves that f decreases with respect to x_1. The right bound is also eliminated in the same way. Then the domain of x_2 is contracted by $\partial\Gamma_2$. It comes the new domain $[-5.250, 8.250]$. Once again, the initial bounds are removed. Similarly, the domain of x_3 is reduced by $\partial\Gamma_3$ to the interval $[-1.625, 5.125]$ and the bounds vanish. Now the box is an interior box and the propagation algorithm iterates, deriving a precise enclosure of the global minimizer $x^* = (3, 4, 3)$.

The second problem [26] corresponds to the function

$$f(x_1, x_2) = x_1^2 \exp(x_2) - x_2^3$$

and the box $\Omega = [-\infty, +\infty] \times [-5, 5]$. In particular, the domain of x_1 is unbounded and the gradient is nonlinear. An evaluation of f at the midpoint of Ω derives the first upper bound $U = 0$. Then the following domain contractions are calculated by Algorithm 2. First, Γ_0 returns the new domains $[-136.3, 136.3]$ for x_1 and $[0, 5]$ for x_2. Then the domain of x_1 is reduced to 0 by $\partial\Gamma_1$. Next, the domain of x_2 is reduced to 0 by $\partial\Gamma_2$ but the right bound $x_2 = 5$ remains since f decreases. The box is then bisected and the global minimizer $x^* = (0, 5)$ is easily found.

The third problem represents a concave function defined as the opposite of the trid function with 3 variables. Let $U = -3$ be the first upper bound. First, Γ_0 has no effect. Then $\partial\Gamma_1$ reduces the domain of x_1 to the interval $[-3.5, 5.5]$ but the initial bounds $x_1 = -9$ and $x_1 = 9$ are not discarded. For similar reasons, the domains of x_2 and x_3 are not modified. Then the domain of x_1 is bisected. The next box $[-9, 0] \times [-9, 9] \times [-9, 9]$ is not contracted. Then the domain of x_2 is bisected. The next box $[-9, 0] \times [0, 9] \times [-9, 9]$ is reduced as an empty box by $\partial\Gamma_1$. Since the function is increasing at the left bound with respect to x_1 then we fix $x_1 = -9$. The domains of the other variables are not modified. After one more bisection, the next box $-9 \times [0, 9] \times [-9, 0]$ is reduced to the global minimizer $x^* = (-9, 9, -9)$.

3.4. *Contraction techniques for interior boxes*

The local optimality condition (10) implies that the gradient must vanish in an interior box. It follows that the contraction algorithm can be implemented by a constraint propagation algorithm applying the contractors $\Gamma_0, \Gamma_1, \ldots, \Gamma_n$. In the following, we will consider the hc4 algorithm and the bc4 algorithm. Moreover, every hc4 contractor must act on the DAG representation of the constraint system, which reduces the localiy effect as previously said.

For example, the bard problem [9] has an objective function defined by a sum of 15 square terms of the form

$$\left(a_i - \frac{1}{b_i x_2 + c_i x_3} - x_1\right)^2$$

given an initial box $\left[-10^3, 10^3\right] \times \left[0, 10^3\right] \times \left[0, 10^3\right]$. This problem is not solvable in reasonable time with hc4 contractors. Using bc4 contractors leads to the enclosure $[0.0082143, 0.0082149]$ of f^*, which takes less than one second and about 10^3 boxes generated in the branch-and-contract algorithm.

The square system of equations $g_1(x) = \cdots = g_n(x) = 0$ can be handled by a multidimensional interval Newton operator [21] in order to contract a given box. The classical one generates a sequence of non-convex linear relaxations by means of the mean value extension of the gradient taken the midpoint of the box as expansion point. Every relaxation is solved by an interval linear method such as the preconditionned interval Gauss-Seidel method. This operator is able to converge quadratically around a solution but it runs in $O(n^3)$ due to matrix inversions. The X-Newton one [2] produces a sequence of convex relaxations from two mean value extensions of the gradient taken opposite corners of the box as expansion points. Every relaxation is processed by $2n$ Simplex algorithms, one per bound of the box. For example the Neumaier3 problem [9]

$$\min \sum_{i=1}^{10}(x_i - 1)^2 - \sum_{i=1}^{9} x_i x_{i+1},\ -100 \leq x_i \leq 10 \quad (i = 1, \ldots, 10)$$

can be solved without branching using the classical Newton operator applied to the box returned by Algorithm 1. This strategy leads to a very precise enclosure of the global minimum $f^* = -210$ in a few milliseconds.

3.5. *Holes and bounds*

Our contractor for boundary boxes implements a specific strategy. Many other strategies could be devised. Puranik and Sahinidis [26] exploit the optimality conditions as done in Algorithm 2 and they use a probing strategy to prune the boundary of the initial box. Given a box X, their algorithm derives a new box Z in three steps with respect to $g_i(x) = 0$ as follows.

(1) $Z \leftarrow \Gamma_i(X)$;
(2) If $\min X_i = l_i$ and $\min Z_i \neq l_i$ then $Z \leftarrow Z \cup \Gamma_0(X^{i-})$;
(3) If $\max X_i = u_i$ and $\max Z_i \neq u_i$ then $Z \leftarrow Z \cup \Gamma_0(X^{i+})$.

A bound of the initial box that is removed by Γ_i is handled by applying the contractor Γ_0 associated with the upper bound constraint to the corresponding facet [29]. With respect to Algorithm 2, there are several differences. When f is decreasing on X^{i-} (or increasing on X^{i+}) the monotonicity test must succeed and the facet is eliminated while Γ_0 does not necessarily discard it, especially when the upper bound

constraint has a complex expression. When f is increasing on X^{i-} (or decreasing on X^{i+}), the facet is left unchanged by Algorithm 2 while it may be contracted by Γ_0. Finally, their algorithm uses only a form of hc4 contractors while Algorithm 2 is parameterized by contractors.

Batnini and Rueher [3] proposed to store the holes in domains generated during the propagation process and to exploit them in further branching steps. Algorithm 2 is able to derive a union of boxes of the form $X^{i-} \cup \Gamma_i(X) \cup X^{i+}$, which means that the domain of x_i may be a union $\{l_i\} \cup Y_i \cup \{u_i\}$ such that Y_i is the i-th component of $\Gamma_i(X)$. It is then possible to generate three nodes in the search tree, one per component of the union. Our tests show that this strategy does not improve much the solving process in general.

Schichl *et al.* [30] introduced an arithmetic of unions of intervals. Such domains can be propagated through the constraints, which leads to more precise domains. There is an additional cost but they suggest to bound the number of intervals in a union by filling some holes. In our context, this technique could be useful to propagate domains of the form $\{l_i\} \cup Y_i \cup \{u_i\}$. Testing this approach requires the development of a new solver, which is postponed for future work.

Kearfott [17] proposed to handle the boundary of the initial box following a peeling strategy. For instance, an initial domain $[l_i, u_i]$ can be divided in three parts $[l_i, l_i + \epsilon]$, $[l_i + \epsilon, u_i - \epsilon]$ and $[u_i - \epsilon, u_i]$ given some tolerance $\epsilon > 0$. It is then possible to generate three new nodes in the search tree. The domain in one node $[l_i + \epsilon, u_i - \epsilon]$ is no more on the boundary, the goal being to early derive interior boxes in order to apply the multidimensional interval Newton operator. The other two domains $[l_i, l_i + \epsilon]$ and $[u_i - \epsilon, u_i]$ are expected to be small enough in order to eliminate the dependency problem stemming from the multiple occurrences of x_i in the constraint system. However, this implies a combinatorial explosion of the number of boxes in general. Our algorithm based on contractors is more constructive.

4. Experimental results

4.1. *Implementation and benchmarks*

The new contractor for boundary boxes $\partial\Gamma_i$ has been integrated into the interval branch-and-contract algorithm of Realpaver [12]. In this software, a bound-constrained optimization problem is represented by a computational graph including the objective function, its gradient and the associated constraints. The second-order derivatives are computed by numerical differentiation [28]. Every box is contracted by Algorithm 1 with hc4 or bc4 contractors (Γ_i and $\partial\Gamma_i$ contractors) followed by the interval Newton operator applied to the gradient constraint $g(x) = 0$. Several interval extensions are used to calculate bounds of the objective function. The interval arithmetic layer uses the elementary functions with correct rounding from the MPFR library [10]. It follows that the interval computations in Realpaver are rigorous. An enclosure of the global optimum is always derived but it may be hard to prove global optimality.

All experiments were conducted on a 2.90GHz Intel Core i7 with 32 GB of RAM running Ubuntu 18.04. In order to measure the impact of $\partial\Gamma_i$, more than 100 problems with at most 10^4 variables have been extracted from [9,16,26]. During the search, a variable domain can be split only if its width is greater than $\epsilon_x = 10^{-8}$. An execution is stopped after a time-out (TO) of 500 seconds if the width of the enclosure of the global minimum remains greater than $\epsilon_f = 10^{-4}$.

4.2. *Results*

The preprocessing phase uses the monotonicity test in order to reduce the problem dimension. For instance, the *pentdi* problem has 10^4 variables. Half of the variables are fixed in this phase. The preprocessing phase including the parsing of the 64 kB input file takes about 0.85 seconds, which represents 43% of the computational time.

In the following, we will compare five strategies that use different contractors and different techniques for boundary boxes.

- S1 applies a pure monotonicity test to boundary boxes such that a variable is fixed if the function is proved to be monotone. Interior boxes are handled by constraint propagation with hc4 contractors and the interval Newton operator.
- S2 corresponds to S1 such that boundary boxes are handled by a constraint propagation algorithm using contractors based on the BARON test [26].
- S3 corresponds to S1 such that boundary boxes are contracted by Algorithm 2.
- S4 corresponds to S3 without the interval Newton operator.
- S5 corresponds to S4 with bc4 contractors instead of hc4 contractors.

A performance profile is depicted in Fig. 5. All the timings include the preprocessing phase, the solving phase and the postprocessing phase. Comparing S3 with S1 and S2 shows that our new algorithm is the best one for boundary boxes. S4 is better than S3 since the interval Newton operator is too expensive for some large scale problems. S5 is the best strategy, which comes from the use of bc4 contractors that are stronger than hc4 contractors. Table 1 shows for each strategy the number of solved problems before the time-out and the number of problems for which the strategy is the best one. For instance, S1 is the best strategy for 32 problems, mainly because these problems are easy to solve, the other strategies doing more work.

Table 1. Comparison of the five strategies.

	S1	S2	S3	S4	S5
solved	95	98	103	105	109
best	32	12	17	24	25

Table 2 reports the results obtained from our strategy S5 and the BARON solver. Each row corresponds to one problem with n variables and *occ* occurrences of variables in the objective function. The number of nodes in the branch-and-contract algorithm and the computation time are reported for both solvers, the results for

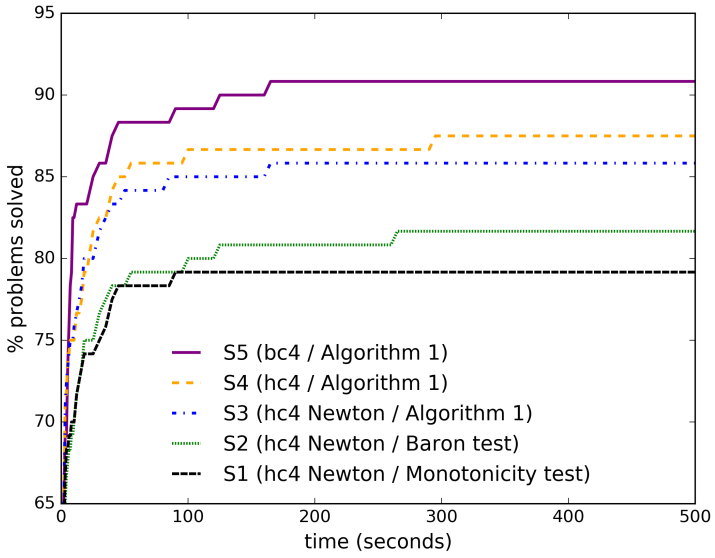

Fig. 5. Performance profile for the five strategies showing the percentage of problems solved as a function of time.

BARON being extracted from [26]. If we compare the numbers of nodes, Realpaver seems very efficient for several problems such as *hosaki*, *ex_4_1_5*, *hartmann3* and *himmelp1*. BARONb exhibits high performances for the least squares problems *model16*, *kowalik* and *s266*. We may conclude that Realpaver competes well with BARON but there is clearly a room for improvement.

Table 3 reports the results obtained from the strategies S3, S4 and S5 for large scale problems. It is clear that Newton is prohibitive for *white_holst*, *pentdi*, *nondia* and *chung_reynolds*, since S3 leads to a time-out and S4 not. However, the Newton operator allows S3 to easily solve the *trid* problem, which is a convex function having only one stationary point. Using bc4 contractors is expensive for *white_holst* since S4 is much better than S5. S5 is clearly the best strategy since the number of nodes is much smaller in general. In particular, the problems *ext_tet*, *engval1* and *dixmaana* are solved only with bc4 contractors.

Problems that are not solved by Realpaver typically have a huge number of occurrences of variables, in particular least squares problems such as *weibull3* or *tranter*. Interval computations are very weak due the dependency problem of interval arithmetic. It could be useful to try other contraction techniques such as strong consistency techniques, for instance a form of strong box consistency [25] or the constructive interval disjunction method [34].

Table 2. Comparison of Realpaver and BARON (results obtained from different machines).

Problem	n	*occ*	RealpaverS5		BARONb	
			nodes	time	nodes	time
ex_8_1_4	2	6	7	0.01	11	0.05
paviani	10	30	5	0.08	1	0.03
hs110	10	30	5	0.08	1	0.11
expquad	120	576	98	0.30	5	2.56
qrtquad	120	575	142	0.16	139	26.95
hosaki	2	6	130	0.03	915	0.94
ex_4_1_5	2	6	7	0.01	3 541 943	500
hartmann3	3	12	13	0.02	179	0.42
himmelp1	2	23	528	0.17	5 357	6.55
s204	2	24	63	0.01	37	0.13
gold	2	16	704	0.04	451	0.89
model33	3	89	21	0.83	17	0.61
model16	4	44	7858	4.28	307	9.18
kowalik	4	44	3484	1.86	265	9.24
stattools	2	20	254	1.03	858	2.72
s266	5	350	23 691	37.20	1 689	5.44

Table 3. Solving large scale problems with Realpaver using the strategies S3, S4 and S5 all based on Algorithm 2.

Problem	n	S3		S4		S5	
		nodes	time	nodes	time	nodes	time
white_holst	500	176	TO	751	**26.81**	497	161.90
trid	20	1	**0.01**	19 830	12.09	19 830	2.80
pentdi	1000	39	TO	39	2.27	39	**2.21**
nondquar	10	67 178	25.12	67 178	24.6	9 979	**6.197**
nondia	999	12	TO	1 006	50.20	1 007	**41.42**
ext_tet	500	4 573	TO	4 591	TO	1	**5.69**
expquad	120	149	0.31	149	**0.30**	98	**0.30**
explin	120	2 485	2.92	2 485	2.95	289	**0.90**
engval1	500	5 409	TO	5 270	TO	13	**1.58**
dixmaana	3000	1318	TO	1 352	TO	1	**22.83**
diagonal1	500	1	2.96	1	**0.21**	1	**0.21**
chung_reynolds	500	1110	TO	6 810	292.10	4	**4.27**

5. Conclusion

In this paper, we have introduced a new interval contraction method for solving bound-constrained global optimization problems. This method exploits the first-order optimality conditions in order to efficiently prune boundary boxes, in two steps. First, an interval contractor is used to enclose the set of stationary points. This contractor can be chosen according to the problem difficulty and we have implemented and tested various forms of consistency techniques, namely hc4 and bc4 contractors. Second, accurate interval-based monotonicity tests are used to reject the initial bounds. The experimental study shows that this new method permits to solve several difficult problems with a lot of variables. Moreover, it competes well

with the probing strategy [26] implemented in the global solver BARON. This study also shows that there is a room for improvement since our solver may suffer from pessimistic interval computations.

References

1. K. R. Apt. The essence of constraint propagation. *Theoretical Computer Science*, 221:179–210, 1999.
2. I. Araya, G. Trombettoni, and B. Neveu. A contractor based on convex interval Taylor. In N. Beldiceanu, N. Jussien, and E. Pinson, editors, *Proceedings of the 9th International Conference on Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimization Problems (CP-AI-OR)*, volume 7298 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2012.
3. H. Batnini, C. Michel, and M. Rueher. Mind the gaps: A new splitting strategy for consistency techniques. In P. van Beek, editor, *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP)*, volume 3709 of *Lecture Notes in Computer Science*, pages 77–91. Springer, 2005.
4. F. Benhamou, F. Goualard, L. Granvilliers, and J.-F. Puget. Revising hull and box consistency. In D. De Schreye, editor, *Proceedings of the 16th International Conference on Logic Programming (ICLP)*, pages 230–244. MIT Press, 1999.
5. F. Benhamou, D. A. McAllester, and P. Van Hentenryck. CLP(Intervals) revisited. In M. Bruynooghe, editor, *Proceedings of the 1994 International Symposium on Logic Programming (ILPS)*, pages 124–138. MIT Press, 1994.
6. G. Chabert and L. Jaulin. Contractor programming. *Artificial Intelligence*, 173(11):1079–1100, 2009.
7. T. Csendes. New subinterval selection criteria for interval global optimization. *Journal of Global Optimization*, 19(3):307–327, 2001.
8. T. Csendes and D. Ratz. Subdivision direction selection in interval methods for global optimization. *SIAM Journal on Numerical Analysis*, 34:922–938, 1996.
9. The Optimization Firm. Bound-constrained programs. http://minlp.com/nlp-and-minlp-test-problems.
10. L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. MPFR: a multiple-precision binary floating-point library with correct rounding. *ACM Transactions on Mathematical Software*, 33(2), 2007.
11. L. Granvilliers. A new interval contractor based on optimality conditions for bound constrained global optimization. In L. H. Tsoukalas, E. Grégoire, and M. Alamaniotis, editors, *Proceedings of the IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 90–97. IEEE, 2018.
12. L. Granvilliers and F. Benhamou. Algorithm 852: RealPaver: An interval solver using constraint satisfaction techniques. *ACM Transactions on Mathematical Software*, 32(1):138–156, 2006.
13. E. R. Hansen. Global optimization using interval analysis – the multi-dimensional case. *Numerische Mathematik*, 34:247–270, 1980.
14. E. R. Hansen and R. I. Greenberg. An interval Newton method. *Applied Mathematics and Computation*, 12:89–98, 1983.
15. P. Van Hentenryck. A gentle introduction to numerica. *Artificial Intelligence*, 103:209–235, 1998.
16. M. Jamil and X. Yang. A literature survey of benchmark functions for global optimization problems. *Int. Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013.

17. R. B. Kearfott. *Rigorous Global Search: Continuous Problems*. Kluwer, Dordrecht, 1996.
18. O. Lhomme. Consistency techniques for numeric CSPs. In R. Bajcsy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intellence (IJCAI)*, pages 232–238. Morgan Kaufmann, 1993.
19. Y. Lin and M. A. Stadtherr. Advances in interval methods for deterministic global optimization in chemical engineering. *Journal of Global Optimization*, 29:281–296, 2004.
20. A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.
21. R. E. Moore. *Interval Analysis*. Prentice-Hall, 1966.
22. R. E. Moore. On computing the range of a rational function of n variables over a bounded region. *Computing*, 16(1):1–15, 1976.
23. A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 13:271–369, 2004.
24. B. Neveu, G. Trombettoni, and I. Araya. Node selection strategies in interval branch and bound algorithms. *Journal of Global Optimization*, 64(2):289–304, 2016.
25. J.-F. Puget and P. Van Hentenryck. A constraint satisfaction approach to a circuit design problem. *J. Global Optim.*, 13:75–93, 1998.
26. Y. Puranik and N. V. Sahinidis. Bounds tightening based on optimality conditions for nonconvex box-constrained optimization. *Journal of Global Optimization*, 67(1-2):59–77, 2017.
27. Y. Puranik and N. V. Sahinidis. Domain reduction techniques for global NLP and MINLP optimization. *Constraints*, 22(3):336–376, 2017.
28. L. B. Rall and G. F. Corliss. Automatic differentiation: Point and interval. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*, pages 165–170. Springer US, Boston, MA, 2009.
29. N. V. Sahinidis. Personal communication, 2018.
30. H. Schichl, F. Domes, T. Montanher, and K. Kofler. Interval unions. *BIT Numerical Mathematics*, 57(2):531–556, 2017.
31. H. Schichl and A. Neumaier. Interval analysis on directed acyclic graphs for global optimization. *Journal of Global Optimization*, 33(4):541–562, 2005.
32. S. Skelboe. Computation of rational functions. *BIT*, 14:1974, 87–95.
33. M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103:225–249, 2005.
34. G. Trombettoni and G. Chabert. Constructive interval disjunction. In *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming (CP)*, volume 4741 of *Lecture Notes in Computer Science*, pages 635–650. Springer, 2007.