J Glob Optim (2008) 42:369–384
DOI 10.1007/s10898-008-9297-7

Efficient interval partitioning for constrained global optimization

Chandra Sekhar Pedamallu · Linet Özdamar · Tibor Csendes · Tamás Vinkó

Received: 7 March 2008 / Accepted: 8 March 2008 / Published online: 27 March 2008

Abstract A new efficient interval partitioning approach to solve constrained global optimization problems is proposed. This involves a new parallel subdivision direction selection method as well as an adaptive tree search. The latter explores nodes (intervals in variable domains) using a restricted hybrid depth-first and best-first branching strategy. This hybrid approach is also used for activating local search to identify feasible stationary points. The new tree search management technique results in improved performance across standard solution and computational indicators when compared to previously proposed techniques. On the other hand, the new parallel subdivision direction selection rule detects infeasible and suboptimal boxes earlier than existing rules, and this contributes to performance by enabling earlier reliable deletion of such subintervals from the search space.

Keywords Constrained global optimization · Interval partitioning · Adaptive search tree management · Subdivision direction selection rules · Parsing

C. S. Pedamallu
School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore, Singapore
e-mail: pes.murali@gmail.com

C. S. Pedamallu
New England Biolabs Inc., 240 County Road, Ipswich, MA, USA

L. Özdamar
Department of Systems Engineering, Yeditepe University, Istanbul, Turkey
e-mail: linetozdamar@lycos.com

T. Csendes (✉)
Institute of Informatics, University of Szeged, Szeged, Hungary
e-mail: csendes@inf.u-szeged.hu

T. Vinkó
ESA/ESTEC, Advanced Concepts Team, Noordwijk, The Netherlands
e-mail: Tamas.Vinko@esa.int

1 Introduction

Many important real world problems can be expressed in terms of a set of nonlinear constraints that restrict the real domain over which a given performance criterion is optimized, that is, as a Constrained Optimization Problem (COP). A COP is defined by an objective function, $f(x_1, \ldots, x_n)$ to be maximized over a set of variables, $V = x_1, \ldots, x_n$, within respective intervals: $X_i = [\underline{X}_i, \overline{X}_i]$ for x_i, $i = 1, \ldots, n$, that are restricted by a set of constraints represented as

$$g_i(x_1, \ldots, x_n) \leq 0 \quad i = 1, \ldots, k,$$

$$h_i(x_1, \ldots, x_n) = 0 \quad i = k + 1, \ldots, r.$$

These constraints can be linear and nonlinear equations or inequalities. An optimal solution of a COP is an element x^* of the search space $X = X_1 \times \cdots \times X_n$, that meets all the constraints, and whose objective function value, $f(x^*) \in f(x)$ for all other consistent elements $x \in X$.

It is hard to tackle the general non-convex COP, and in general, traditional numeric algorithms cannot guarantee global optimality and completeness in the sense that the solution found might be only a local optimum. Here, we introduce an Interval Partitioning Algorithm (IP) that subdivides the continuous domain over which the COP is defined and conducts reliable assessment of the subdomains (boxes) while searching for the globally optimal solution. The reliability is meant in the sense that a box that has a potential to contain a global optimizer point is never discarded. By principle, an interval partitioning method continues to subdivide a given irrelevant box until either it turns out to be infeasible or suboptimal. Then, the box is discarded by the feasibility or optimality cut-off tests. Otherwise it becomes a small enclosure (by nested partitioning) which possibly contains a local stationary point. During the partitioning process, an increasing number of solution candidates are identified by invoking a local search procedure in promising boxes. Hence, similar to other interval and non-interval B&B techniques, a local search procedure is utilized in IP to find x^* in a given box. Here, we use Feasible Sequential Quadratic Programming, FSQP, as a local search method that has convergence guarantee when started from a location nearby a stationary point.

Theoretically, IP has no difficulties in dealing with the COP; however, interval research on the COP is relatively scarce when compared with bound constrained optimization. An early reference is that of Robinson who used interval arithmetic to obtain bounds for the solution of the COP [31]. Hansen and Sengupta [14] first used IP to solve the inequality COP. A detailed discussion on interval techniques for the general COP with both inequality and equality constraints is provided in Ratschek and Rokne [29] and Hansen [13], and some numerical results using these techniques have been published later in [36]. Dallwig et al. [9] proposed the software GLOPT for solving bound constrained optimization and the COP and a new reduction technique was suggested. Kearfott presented GlobSol, which is an IP software that is capable of solving bound constrained optimization problems and the COP [16]. Markót [20] developed an IP for solving the COP with inequalities where new adaptive multi-section rules and a new box selection criteria were presented [21].

Our contribution described in Sect. 2 to the generic IP lies in two features: a new adaptive tree search method that can be used both in non-interval and interval B&B approaches, and a new subdivision direction selection (branching) rule that can be used in interval methods. This new branching rule aims at reducing the uncertainty degree in the feasibility regarding the constraints over a given subdomain as well as the uncertainty in the box's potential of containing a global optimizer point. The new techniques are based on symbolic manipulation of the expressions [6,23,24,28]. While in [6,28] nonlinear transformation was applied to simplify the objective function, in [23,24] the parsing of the computation tree was introduced to have a good decision on the subdivision directions.

Our numerical experiments (in Sect. 4) show that the resulting IP is a viable method for these problems. The results are compared with commercial software such as BARON, MINOS, and other solvers interfaced with GAMS.

2 Interval partitioning algorithm for the COP

The set of compact intervals is denoted by $\mathbb{I}$. Every interval $X \in \mathbb{I}$ is also denoted by $[\underline{X}, \overline{X}]$, with the bounds of $\underline{X} = \min X$ and $\overline{X} = \max X$. For every $a \in R$, the *point interval* $[a, a]$ is also denoted by a. The *width* of an interval X is the real number $w(X) = \overline{X} - \underline{X}$. Given two real intervals X and Y, X is said to be tighter than Y if $w(X) < w(Y)$.

Elements of $\mathbb{I}^n$ are the boxes. Given $X_1, \ldots, X_n \in \mathbb{I}$, the corresponding *box* X is the Cartesian product of intervals, $X = X_1 \times \cdots \times X_n$, where $X \in \mathbb{I}^n$. A subset of X, $Y \subseteq X$, is a subbox of X. The notion of width is defined as

$$w(X_1 \times \cdots \times X_n) = \max_{1 \le i \le n} w(X_i).$$

Interval arithmetic operations are set theoretic extensions of the corresponding real operations [1]. Given $x, y \in \mathbb{I}$, and an operation $\lozenge \in \{+, -, \times, \div\}$, we have: $X \lozenge Y = \{x \lozenge y \mid x \in X, y \in Y\}$. Due to properties of monotonicity, these operations can be implemented by real computations over the bounds of intervals. The associative law and commutativity hold for these operations, but the distributive law does not hold (only a weaker law is true, subdistributivity).

Interval arithmetic is particularly appropriate to represent inclusion of real quantities. The range of a real function f over an interval X is denoted by $f(X)$, and it can be computed by interval extensions.

Definition 1 *An inclusion function of a real function* $f\colon D_f \subset R^n \to R$ *is a function* $F\colon \mathbb{I}^n \to \mathbb{I}$ *such that* $\forall X \in \mathbb{I}^n, X \in D_f \Rightarrow f(X) = \{f(x) \mid x \in X\} \subseteq F(X)$.

This inclusion formula is the basis of what is called the *fundamental theorem of interval arithmetic*: interval extensions always produce inclusion functions that enclose the range of the corresponding real function. As a result, suppose that, for instance, you are looking for a zero of a real function f over a domain D. If the evaluation of an interval extension of f over D does not contain 0, it means that 0 was not part of the range of f over D. In a proper computer implementation of interval extension based inclusion functions the outside rounding must be made to have guaranteed reliability.

The most common interval extension is known as *natural extension*. This is obtained by replacing each arithmetic operation and standard function found in the expression of a real function with an enclosing interval operation. Natural extension functions are *inclusion isotone*: given a real function f, whose natural extension is denoted by F, and two intervals X and Y such that $X \subseteq Y$, the following holds: $F(X) \subseteq F(Y)$. For real operations this property follows from their monotonicity. We denote the lower and upper bounds of the function interval range over a given box Y by $\underline{F}(Y)$ and $\overline{F}(Y)$, respectively.

Here, it is assumed that for the studied COP, the natural interval extensions of f, g, and h over X are defined. Furthermore it is assumed that, F (and similarly, G and H) is α-*convergent* over X, that is, there exist $c, \alpha > 0$ such that $w(F(Y)) - w(f(Y)) \le cw(Y)^{\alpha}$ holds for all $Y \subseteq X$.

An *interval constraint* is built from an atomic interval formula (interval function) and relation symbols, whose semantics are extended to intervals as well. A constraint is defined by its expression (atomic formula and relation symbol), its variables, and their domains. We will consider that an interval constraint has interval variables (variables that take interval values), and that each associated domain is an interval.

The main feature of interval constraints is that if its solution set is empty, it has no solution over a given box Y then it follows that the solution set of the COP is also empty and the box Y can be reliably discarded. Suppose the objective function value of a feasible solution is known as the Current Lower Bound, CLB. Then, similar to infeasible boxes, suboptimal boxes can be discarded as follows. If the upper bound of the objective function range over a given box Y is less than CLB, then Y can be discarded since it cannot contain a better solution than the CLB.

Below we formally provide the conditions when a given box Y can be discarded reliably, based on the ranges of interval constraints and the objective function. In a partitioning algorithm, each box Y is assessed for its optimality and feasibility status by calculating the inclusions F, G, and H over the domain of Y.

Definition 2 (*Cut off test based on optimality:*) *If* $\overline{F}(Y) < CLB$, then the box Y is called a *suboptimal* box.

Definition 3 (*Cut off test based on feasibility:*) *If* $\underline{G}_i(Y) > 0$, *or* $0 \notin H_i(Y)$ for any i, then box Y is called an *infeasible* box.

Definition 4 If $\underline{F}(Y) \leq CLB$, and $\overline{F}(Y) \geq CLB$, then Y is called an *indeterminate* box *with regard to optimality.* Such a box holds the potential of containing x^* if it is not an infeasible box.

Definition 5 If ($\underline{G}_i(Y) < 0$, and $\overline{G}_i(Y) > 0$), or ($0 \in H_i(Y) \neq 0$) for some i, and other constraints are consistent over Y, then Y is called an *indeterminate box with regard to feasibility* and it holds the potential of containing x^* if it is not a suboptimal box.

Definition 6 *The* degree of uncertainty *of an indeterminate box* with respect to optimality *is defined as:* $PF(Y) = \overline{F}(Y) - CLB$.

Definition 7 *The* degree of uncertainty, $PG^i(Y)$ ($PH^i(Y)$) *of an indeterminate inequality* (*equality*) *constraint* with regard to feasibility *is as:* $PG^i(Y) = \overline{G_i}(Y)$, *and* $PH^i(Y) = \overline{H_i}(Y) + |\underline{H}_i(Y)|$.

Definition 8 *The* total feasibility uncertainty degree *of a box,* $INF(Y)$, *is the sum of uncertainty degrees of equalities and inequalities that are indeterminate over* Y.

The new subdivision direction selection rule (Interval Inference Rule, IIR) targets an immediate reduction in $INF(Y)$ and $PF(Y)$ and chooses the respective variables to bisect a given parent box. The IP described in the following section uses the feasibility and optimality cut-off tests in discarding boxes and applies the new rule IIR in partitioning boxes.

2.1 Interval partitioning algorithm

IP is an algorithm that subdivides indeterminate boxes to reduce $INF(Y)$ and $PF(Y)$ by nested partitioning. The contraction and α-convergence properties enable this. The reduction in the uncertainty levels of boxes finally lead to their elimination due to suboptimality or infeasibility while helping IP in ranking the remaining boxes in a better way.

Algorithm 1 Interval partitioning procedure

```
1. Step 0. Set the initial box $Y = X$ and the list of indeterminate boxes, $B = \{Y\}$.
2. Step 1. If $B = \emptyset$, or if the number of function calls or CPU time reaches a given limit, then STOP.
   Else, select the first box, $Y$ in $B$ and remove it from the list.
   If $Y$ is infeasible or suboptimal, go to Step 1.
   If $Y$ is sufficiently small in width, evaluate $m$, its mid-point, and if it is a feasible improving solution, update
   CLB, and go to Step 1.
3. Step 2. Select coordinate directions to partition (Use the subdivision direction selection rule IIR). Set $v$ to
   the number of coordinates to subdivide.
4. Step 3. Partition $Y$ into $2^v$ non-overlapping child boxes and add them to $B$. Go to Step 1.
```

A box that has no uncertainty with regard to feasibility after nested partitioning still has uncertainty with regard to optimality unless it is proven that it is suboptimal. The convergence rate of IP might be slow if we require nested partitioning to reduce a box to a point interval that is the global optimizer point. Hence, since a box with a high $PF(Y)$ is promising that it may contain a global optimizer, we use a local search procedure that can identify stationary points in such boxes.

Usually, IP continues to subdivide available indeterminate and feasible boxes until either they are all deleted or interval sizes of all variables in existing boxes are less than a given tolerance. Such boxes may contain a global optimizer point. Termination can also be forced by limiting the number of function evaluations and/or CPU time. Here, we choose to terminate IP when the number of function calls not counting those of the local search procedure reaches a given limit or when the CPU time exceeds a maximal allowed value. Algorithm 1 is a generic IP algorithm without calls to local search.

We now describe our new IP that has a flexible stage-wise tree management feature. This stage-wise search tree applies the best-first box selection rule within a restricted subtree (to economize memory usage), meanwhile it invokes local search in a set of boxes.

The tree management system in the proposed IP maintains a stage-wise branching scheme that is conceptually similar to the iterative deepening approach [19]. The iterative deepening approach explores all nodes generated at a given search tree level (stage) before it starts assessing the nodes at the next stage. Exploration of boxes at the same stage can be done in any order, the sweep may start e.g. from best-first box of that stage. On the other hand, in the proposed adaptive tree management system, a node (parent box) at the current stage is permitted to grow a subtree forming partial succeeding tree levels and to explore nodes in this subtree before exhausting the nodes at the current stage.

If a feasible solution (and CLB) is not identified yet, boxes in the subtree are ranked according to descending $INF(Y)$ values, otherwise they are ranked in descending order of $\overline{F}(Y)$. A box is selected among the children of the same parent according to either box selection criterion, and the child box is partitioned again continuing to build the same subtree. This subtree grows until the Total Area Deleted (TAD) by discarding boxes fails to improve in two consecutive partitioning iterations in this subtree. Such failure triggers a call to local search where all boxes not previously subject to local search are processed by the procedure Feasible Sequential Quadratic Programming (FSQP, [38]), after which they are placed back in the list of pending boxes and exploration is resumed among the nodes at the current stage. If a feasible solution with a better objective function value is found by FSQP the CLB is updated and the solution is stored.

The above adaptive tree management scheme maintains two lists of boxes, B_s and B_{s+1} that are the lists of boxes to be explored at the current stage s and the next stage $s+1$, respectively. Initially, the set of indeterminate or feasible boxes in the pending list B_s consists only

Algorithm 2 Interval partitioning with adaptive tree management

```
1. Step 0. Set tree stage, $s = 1$. Set future stage, $r = 1$. Set non-improvement counter for TAD: $nc = 0$. Set
   $B_s = \{X\}$, and $B_{s+1} = \emptyset$.
2. Step 1. If the number of function evaluations or CPU time reaches the given limit, or, both $B_s = \emptyset$ and
   $B_{s+1} = \emptyset$, then STOP.
   Else, if $B_s = \emptyset$ and $B_{s+1} \neq \emptyset$, then set $s \leftarrow s + 1$, set $r \leftarrow s$, and continue. Select the first box $Y$ in $B_s$
   and remove it from $B_s$.
   1.1 If $Y$ is infeasible or suboptimal, go to Step 1.
   1.2 Else if $Y$ is sufficiently small, evaluate $m$, its mid-point, and if it is a feasible improving solution, update
   CLB, reset $nc \leftarrow 0$, and store $m$. Go to Step 1.
3. Step 2. Select variable(s) to partition (use the subdivision direction selection rule IIR). Set $v$ equal to the
   number of variables to partition.
4. Step 3. Partition $Y$ into $2^v$ non-overlapping child boxes. Check TAD, if it improves, then reset $nc \leftarrow 0$,
   else set $nc \leftarrow nc + 1$.
5. Step 4. Add the $2^v$ boxes to $B_r$.
   4.1. If $nc > 2$, apply FSQP to all (previously unprocessed by FSQP) boxes in $B_s$ and $B_{s+1}$, reset $nc \leftarrow 0$.
   If FSQP is called for the first time in stage $s$, then set $r \leftarrow s + 1$.
   4.2. Go to Step 1.
```

of X and B_{s+1} is empty. As child boxes are added to a selected parent box, they are ordered according to the current criterion. Boxes in the subtree stemming from the selected parent at the current stage are explored and partitioned until there is no improvement in TAD in two consecutive partitioning iterations.

At that point, partitioning of the selected parent box is stopped and all boxes that have not been processed by local search are sent to the FSQP module and processed to identify feasible and improving point solutions if FSQP is successful in doing so. It should be noted that, whether or not FSQP fails to find an improving solution, IP will continue to partition the box since it passes both cutoff tests as long as it has a potential to contain an improving solution. Finally, the algorithm encloses potential improving solutions in sufficiently small boxes if FSQP can identify them. Thus, FSQP acts as a catalyst that occasionally scans larger boxes to identify improving solutions at the earlier stages of the search. From that moment, child boxes generated from any other selected parent in B_s are stored in B_{s+1} irrespective of further calls to FSQP in the current stage. When all boxes in B_s have been assessed (discarded or partitioned), the search moves to the next stage, $s + 1$, starting to explore the boxes stored in B_{s+1}.

In this manner, less boxes (those in the current stage) are maintained in memory and the search is allowed to go down to deeper levels within the same stage, increasing the chances to discard boxes. On the other hand, by enabling the search to also explore boxes horizontally across at the current stage, it might be possible to find feasible improving solutions faster by not partitioning parent boxes that are not so promising (because we are able to observe a larger number of boxes).

The tree continues to grow in this manner taking the list of boxes of the next stage after the current stage's list of boxes is exhausted. The algorithm stops either when there are no boxes remained in B_s and B_{s+1} or when there is no improvement in CLB as compared with the previous stage. The proposed IP algorithm is given in Algorithm 2. The adaptive tree management system in IP is illustrated in [25].

2.2 A new subdivision direction selection rule for IP

The order in which variable domains are partitioned has an impact on the convergence rate of IP. In general, variable selection is made according to widest coordinate direction rule or largest function rate of change in the box. Here, we develop a new numerical subdivision direction selection rule, Interval Inference Rule (IIR), to improve IP's convergence rate by partitioning in parallel, those variable domains that reduce $PF(Y)$ and $INF(Y)$ in immediate child boxes. Hence, new boxes are formed with an appropriate partitioning sequence resulting in decreased uncertainty caused by the overestimation in the indeterminate objective function range and constraint ranges.

Before IIR is applied, the objective f and each constraint g_i and h_j are interpreted as binary trees that represent recursive subexpressions hierarchically. Such binary trees enable interval propagation over all subexpressions of the constraints and the objective function [3]. Interval propagation and function trees are used by [15] in improving interval Newton approach by decomposition and variable expansion, by [34] in automated problem reformulation, by [32] and by [35] where feasibility based range reduction is achieved by tightening variable bounds.

After interval propagation is carried out over the subexpressions in a binary tree, IIR traverses this tree to label its nodes so as to identify the pair of variables (source variables) that are most influential on the constraint's or the objective's uncertainty degree. The presented interval subdivision direction selection rule is an alternative of earlier rules as those published in [4,8,30]. This pair of variables is identified for each constraint and the objective function, and placed in the pool of variables whose domains will be possibly partitioned in the next iteration. We make sure that the pool at least contains the source variables for the objective function and therefore, the number of variables to be bisected in parallel is at least two. The total pool resulting from the traversal of f, g and h is screened and its size is reduced by allocating weights to variables and re-assessing them.

Before the *labeling* process IIR-Tree can be applied on a constraint expression, it has to be parsed and intervals have to be propagated through all subexpression levels. This is achieved by calling an interval library at each subexpression level of the binary tree from bottom to top starting from atomic levels (variables or constants).

A binary tree representing a constraint is built as follows. Leaves of the binary tree are atomic elements, i.e. they are either variables or constants. All other nodes represent binary expressions of the form (Left Θ Right). A binary operator "Θ" is an arithmetic operator ($\cdot, +, -, \div$) having two branches ("Left" and "Right") that are themselves recursive binary subtrees. However, mathematical functions such as ln, exp, sin, etc. are unary operators. For these the argument of the function is always placed in the "Left" branch. For a detailed example see [24].

When the number of constraints is large, there might be a large set of variables resulting from the application of IIR-Tree to the objective function and each constraint. Here, we develop a priority allocation scheme to narrow down the set of variables (selected by IIR-Tree) to be partitioned in parallel. In this approach, all variable pairs identified by IIR-Tree are merged into a single set Z. Then, a weight w_j is assigned to each variable $x_j \in Z$ and the average $\overline{w}$ is calculated. The final set of variables to be partitioned is composed of the two source variables of f and all other source variables $x_j \in Z$ with $w_j > \overline{w}$ belonging to the constraints.

The weight w_j is defined as a function of several criteria: $PG^i(Y)$ (or PH^i_Y) of constraint g_i (or h_i), for which x_j is identified as a source variable; the number of times x_j exists in the expression of g_i; and the total number of multiplicative terms in which x_j is involved within g_i. Furthermore, the existence of x_j in a trigonometric and/or even power subexpression in g_i is included in w_j by inserting corresponding flag variables. When a variable x_j is a source variable to more than one constraint, the weight calculated for each such constraint is added to result in a total weight w_j defined as

$$\sum_{i\in IC_j} (PF(Y)^i/PH_{\max} + PG(Y)^i/PG_{\max} + e_{ji}/e_{j,\max} + a_{ji}/a_{j,\max} + t_{ji} + p_{ji})/5$$

where IC_j: set of indeterminate constraints (over Y) where x_j is a source variable, TIC: total set of indeterminate constraints, PH_{max}: $\max_{i\in TIC}\{PH^i(Y)\}$, PG_{max}: $\max_{i\in TIC}\{PG^i(Y)\}$, e_{ji}: number of times x_j exists in the constraint number $i \in IC_j$, $e_{j,max}$: $\max_{i\in IC_j}\{e_{ji}\}$, a_{ji}: number of multiplicative terms x_j is involved in constraint $i \in IC_j$, $a_{j,max}$: $\max_{i\in IC_j}\{a_{ji}\}$, t_{ji}: binary parameter indicating that x_j exists in a trigonometric expression in the constraint number $i \in IC_j$, p_{ji}: binary parameter indicating that x_j exists in an even power or abs expression in the constraint number $i \in IC_j$.

This weighting method is illustrated on a collection of constraints that consists of 3 constraints involving 4 variables. The three constraints are given in Eqs. 1–3 below. Variable domains are $X_1 = [-2.0, 4.0]$, $X_2 = [0.0, 10.0]$, $X_3 = [-2.0, 1.0]$, and $X_4 = [-10.0, 0.0]$.

$$1 - (10x_1 + 6x_1x_2 - 6x_3x_4) = 0 \tag{1}$$

$$6(x_1x_4) + 6(x_2x_3) - 10x_3 - 4 = 0 \tag{2}$$

$$\sin(x_1x_2)\cos(x_1^2 - x_2) + (x_1x_4) = 0 \tag{3}$$

In Table 1 we provide a summary of symbolic characteristics for each variable in each constraint. Here TIC is the set of three constraints. The variable weights w_j are calculated using the values of box- and constraint-related parameters given in Table 1. The pairs of maximum impact variables found are (x_1, x_2), (x_1, x_4), and (x_1, x_4) for the first, second and third constraints, respectively. The set Z is $\{x_1, x_2, x_4\}$. A sample weight calculation for x_1 in the first constraint is given as $\left(\frac{600}{600} + \frac{2}{3} + \frac{1}{3} + \frac{0}{1} + \frac{0}{1}\right)/5 = 0.4$. The weight calculations are summarized in Table 2.

Consequently, the pair (x_1, x_4) is selected for re-partitioning. This results in 3 child boxes whose total $INF(Y)$ is indicated by bold face in the Table 3. One child box is found to be infeasible and discarded. For comparison purpose, we also show the total $INF(Y)$ for child boxes that would result from re-partitioning other pairs of variables. It is observed that $INF(Y)$ is the least for the pair (x_1, x_4).

Table 1 Inputs for calculating variable weights

Constraint no.	x_j	e_{ji}	a_{ji}	p_{ji}	t_{ji}	$[\underline{H_i}(Y), \overline{H_i}(Y)]$	PH_Y^i
Constraint 1	x_1	2	1	0	0		
	x_2	1	1	0	0	$[-339, 261]$	$339 + 261 = 600$
	x_3	1	1	0	0		
	x_4	1	1	0	0		
Constraint 2	x_1	1	1	0	0		
	x_2	1	1	0	0	$[-374, 196]$	$374 + 196 = 570$
	x_3	2	1	0	0		
	x_4	1	1	0	0		
Constraint 3	x_1	3	3	1	1		
	x_2	2	2	0	1	$[-41, 21]$	$41 + 21 = 62$
	x_3	0	0	0	0		
	x_4	1	1	0	0		

Table 2 A summary of the calculation of weights for source variables

Variable	Weight in constraint no. 1	Weight in constraint no. 2	Weight in constraint no. 3	Total variable weight (w_j)
x_1	0.4	0.32	0.82	1.54
x_2	0.4	0.59	0.42	1.41
x_4	0.6	0.59	0.42	1.61

Table 3 Total $INF(Y)$ of child boxes resulting from partitioning different pairs of variables

Selected variables	(x_1, x_4)	(x_1, x_2)	(x_1, x_3)	(x_2, x_3)	(x_2, x_4)	(x_3, x_4)
Total $INF(Y)$ of child boxes	**2141**	3088	2758	3786	3778	3728

3 Theoretical results

The inclusion functions applied are obtained by natural interval extension. Hence the inclusion functions have the usual properties [1,29]: they are zero convergent ($\lim_{i\to\infty} w(X_i) = 0$ implies $\lim_{i\to\infty} w(F(X_i)) = 0$), inclusion isotone, and α-convergent for $\alpha = 1$. The parsing is used only to determine the subdivision directions. The decrease in the inclusion function widths achieved within a single subdivision step will be characterized. Only such argument intervals are investigated that have positive width component intervals. The multiplication by the interval [0, 0] is excluded from the forthcoming consideration. We have proven the following four lemmas in [24].

Lemma 1 *Let the operator $\ominus$ at any level k of a binary tree be the m-th power (m is even) or the absolute value, and let $\Lambda^k = \underline{L}^k = 0$. Further, let $\underline{L}^{k+1} < 0$. Then, IIR may not able to identify Λ^{k+1}.*

Lemma 2 *Let* trig *denote any trigonometric function. Define* maxtrig *and* mintrig *as the maximum and the minimum values* trig *can take during one complete cycle. Further, let the operator at any level k of a binary tree be $\ominus$ =* trig, *and* maxtrig $\in [\underline{L}^k, \overline{L}^k] \bigcup \{-\infty, \infty\}$ *or* mintrig $\in [\underline{L}^k, \overline{L}^k] \bigcup \{-\infty, \infty\}$. *Then, IIR may not be able to identify Λ^{k+1}.*

Lemma 3 *Suppose the interval operator at a given level k is $\ominus$ = '$\times$', and $\underline{L}^{k+1}$, $\underline{R}^{k+1} < 0$, $\overline{L}^{k+1}$, $\overline{R}^{k+1} > 0$ $|\underline{L}^{k+1}| = \overline{R}^{k+1}$, and $|\underline{R}^{k+1}| = \overline{L}^{k+1}$. Then, IIR may not be able to label a bound in the right or left subtrees at the level $k + 1$.*

Lemma 4 *For expressions excluding the ambiguous subexpressions indicated in Lemmas 1–3, IIR identifies the correct couple of bounds at level $k+1$ that result exactly in Λ^k at level k.*

Theorem 1 below states that unless ambiguous subexpressions indicated in Lemmas 1–3 exist in a constraint expression or in the objective function f, the partitioning of a parent box Y along a source variable identified by IIR guarantees an immediate reduction in the labeled bound of g_i, h_i, or f. Hence, a reduction will happen in the degrees of uncertainty ($PG^i(Y)$, $PH^i(Y)$, or $PF(Y)$) for at least one immediate child box.

Theorem 1 *Suppose that a given constraint $g_i(x)$ or $h_i(x)$, or an objective function f does not contain the subexpression types indicated in Lemmas 1–3. Assume further that all interval* *components of Y have a positive width and that multiplication by the interval* $[0, 0]$ *is not allowed. Let Y be a parent box that is partitioned by at least one source variable identified by IIR. Then, $PG^i(S_t) < PG^i(Y)$, $PH^i(S_t) < PH^i(Y)$, or $PF(S_t) < PF(Y)$ for at least one child S_t of Y.*

Proof Consider the evaluation of $G_i(Y)$, the cases of the other inclusion functions can be proven similarly. We show that there is an immediate guaranteed reduction in the uncertainty degrees of three children S_t, $t = 1, \dots, 3$, assuming that there are two source variables, x_r, x_m identified by IIR for the box Y and for $g_i(x)$. We define S_1, S_2, S_3, and S_4 as four subboxes produced by the parallel bisection along these source variables. We denote the component intervals belonging to x_r, x_m and box Y as: $Y_r = [\underline{Y}_r, \overline{Y}_r]$ and $Y_m = [\underline{Y}_m, \overline{Y}_m]$, respectively. Variable domains in a given child are denoted by Y_j^S, $j = 1, 2, \dots, n$. Then $Y_j^S = Y_j$, $\forall j \neq r, m$. In Table 4, The r and m coordinate components of the child domains are listed.

Assume that $\overline{Y}_r$ and $\overline{Y}_m$ was identified as the source bounds most contributing to $\overline{G_i}(Y)$. The proof is similar for any other pair of bounds. Below we show that $PG^i(S_t) < PG^i(Y)$ for the children $S_t = S_1$, S_2, and S_3, and that $PG^i(S_4) = PG^i(Y)$.

Case S_1: obviously $S_1 \subseteq Y$. Then, by inclusion isotonicity, $w(G_i(S_1)) \leq w(G_i(Y)))$ and $\overline{G}_i(S_1) \leq \overline{G}_i(Y))$. Further, since $\overline{S}_{1,r} \neq \overline{Y}_r$ and $\overline{S}_{1,m} \neq \overline{Y}_m$, then $\overline{G}_i(S_1) \neq \overline{G}_i(Y)$ due to the fact that $\overline{Y}_r$ and $\overline{Y}_m$ are the source bounds most contributing to $\overline{G_i}(Y)$. This is the point where we utilize the assumptions that all interval components have a positive width and that multiplication by the interval $[0, 0]$ is not allowed.

From the above, $\overline{G}_i(S_1) < \overline{G}_i(Y)$ holds as strict inequality which leads to $PG^i(S_1) < PG^i(Y)$. One can show by similar reasoning that $PG^i(S_2) < PG^i(Y)$ and $PG^i(S_3) < PG^i(Y)$. However, $PG^i(S_4) = PG^i(Y)$, because $\overline{S}_{4,r} = \overline{Y}_r$ and $\overline{S}_{4,m} = \overline{Y}_m$.

The above reasoning is applicable to all bound combinations of contributing source bounds not only for the $(\overline{S}_r, \overline{S}_m)$ pair. In each case, three of the children result in reduced $PG^i(S_t)$ values. When only one source variable is partitioned, the $\overline{Y}_r$ and $\overline{Y}_m$ are the most contributing source bounds to $\overline{G_i}(Y)$ and two children boxes are obtained, then S_1 is guaranteed to have reduced PG^i value compared to that of Y. $\square$

Based on Theorem 1, we now describe supporting rules that are applied by IIR in case labeling ambiguities described in the Lemmas are in the expressions. For subexpressions of the type indicated in Lemma 1 at level k of a binary tree (with $\Lambda^k = \underline{L}^k = 0$ and with such an interval bound at level $k+1$, that $\underline{L}^{k+1} < 0$) set the bound labeling rule to be applied by IIR at level $k+1$ such that $\Lambda^{k+1} = \underline{L}^{k+1}$. That is, $\underline{L}^{k+1}$ is targeted, and hence this ambiguity will be reduced as Theorem 1 proves for other targets. As a consequence, this rule indirectly supports IIR's reduction of $INF(Y)$ or $PF(Y)$.

Assume now that there exist a trigonometric type subexpression at level k of a binary tree with *maxtrig* $\in [\underline{L}^k, \overline{L}^k]$ or *mintrig* $\in [\underline{L}^k, \overline{L}^k]$. Set the bound labeling rule to be applied by IIR at level $k+1$ such that $\Lambda^{k+1} = \max\{|\underline{L}^{k+1}|, |\overline{L}^{k+1}|\}$. The ambiguity at level k

Table 4 Component intervals Y_r^S and Y_m^S for the subboxes discussed in the proof of Theorem 1

Subbox	Y_r^S	Y_m^S
S_1	$[\underline{Y}_r, \underline{Y}_r + w(Y_r)/2]$	$[\underline{Y}_m, \underline{Y}_m + w(Y_m)/2]$
S_2	$[\underline{Y}_r + w(Y_r)/2, \overline{Y}_r]$	$[\underline{Y}_m, \underline{Y}_m + w(Y_m)/2]$
S_3	$[\underline{Y}_r, \underline{Y}_r + w(Y_r)/2]$	$[\underline{Y}_m + w(Y_m)/2, \overline{Y}_m]$
S_4	$[\underline{Y}_r + w(Y_r)/2, \overline{Y}_r]$	$[\underline{Y}_m + w(Y_m)/2, \overline{Y}_m]$

is resolved more and more in forthcoming partitioning iterations when $[\underline{L}^k, \overline{L}^k]$ excludes *maxtrig* or *mintrig*.

For the exceptional case handled in Lemma 3, it is the same which of the two fitting pairs of bounds is selected, the targeted value can occur only in one of its subintervals. These auxiliary rules allow us to handle also expressions involving the problematic operations characterized in Lemmas 1–3. The statements of this section provide only hints that the new direction selection rules can improve the efficiency of the IP algorithm. An extensive computational testing can tell more on that.

4 Numerical tests

We have completed some numerical experiments with IP. First, we compare the performance of different variable selection rules (Rule A, Rule C) from the literature applicable for our IIR too. We also compare the adaptive tree management approach with two conventional tree management approaches: worst-first (we aim at discarding boxes as soon as possible) and depth-first. All three rules and the three tree management techniques are embedded in IP that uses FSQP. We also compare two box ranking approaches, the one with the swap among two criteria (maximal infeasibility $INF(Y)$ and maximal box upper bound on the objective function $\overline{F}(Y)$) and a penalty approach that combines infeasibility with f, namely maximal upper bound of augmented f.

Thus, we shall measure the effectiveness of the method's different features on performance. In order to have a proper background for comparison, we considered the results of five solvers that are linked to the commercial software GAMS and a stand-alone one, FSQP [38] whose code has been provided by AEM (see http://www.aemdesign.com/FSQPmanyobj.htm). The solvers used in this comparison are BARON 7.0 [32], Conopt 3.0 [10], LGO 1.0 [26], MINOS 5.5 [22], Snopt 5.3.4 [12] and FSQP. We allow every solver to complete its run without imposing additional stopping criteria except the limit on CPU time.

4.1 Experimental environment

We made our extensive numerical experiments on a set of 60 COP benchmarks, five of them involving trigonometric functions. Most of these test problems are extracted from the COCONUT benchmark library [5] and that of the Princetonlib [27]. These problems are listed in the Table 5 with the number of dimensions, number of linear and nonlinear inequalities and equalities. While executing IP, we allow at most 2,000 times (the number of variables + the number of constraints) function calls carried out in addition to FSQP calls. The number of iterations allowed for FSQP is limited to 100. We also restrict IP's run time by 900 s (i.e. by 2.827 Standard Time Units as defined in [33]). One STU is equivalent to 318.369 s on our machine. All runs were executed on a PC with 256 MB RAM, 1.7 GHz P4 Intel CPU. The IP was coded in C++ interfaced with the PROFIL interval arithmetic library [18]. In order to illustrate the impacts of IP's individual features (the swapping double box ranking criteria, the adaptive tree search scheme and the branching rule IIR), we compared the following IP variants:

(i) IP with Widest Variable Rule (Rule A [7,8]); IP with Rule C (also called maximum smear, [7,17]); IP with IIR;
(ii) IP with depth-first tree search approach; IP with the best-first tree search approach where box ranking is the same as that of adaptive tree approach; IP with adaptive tree management technique;

Table 5 List of COP benchmarks used in the experiments: problem data and references

Problem	D, NE, LE, NIE, LIE	Source	Problem	D, NE, LE, NIE, LIE	Source
Aircraftb	18, 5, 5, 0, 0	[5]	Hs053	5, 0, 3, 0, 0	[5]
Avgasb	8, 0, 0, 10, 0	[27]	Hs056	7, 4, 0, 0, 0	[5]
Alkyl	14, 6, 1, 0, 0	[5]	Hs407	5, 3, 0, 0, 0	[5]
Bt4	3, 1, 1, 0, 0	[5]	Hs108	9, 0, 0, 12, 0	[5]
Bt8	5, 2, 0, 0, 0	[5]	Hs080	5, 3, 0, 0, 0	[5]
Bt12	5, 3, 0, 0, 0	[5]	Hs043	4, 0, 0, 3, 0	[5]
Bt11	5, 2, 1, 0, 0	[5]	Hs116	13, 0, 0, 10, 5	[5]
Bt7	5, 3, 0, 0, 0	[5]	Himmel11	9, 3, 0, 0, 1	[5]
Dispatch	4, 1, 0, 0, 1	[5]	Immum	21, 0, 7, 0, 0	[5]
Dipigri	7, 0, 0, 4, 0	[5]	Lootsma	3, 0, 0, 2, 0	[5]
Degenlpa	20, 0, 14, 0, 0	[5]	Lewispol	6, 6, 3, 0 ,0	[5]
Degenlpb	20, 0, 15, 0, 0	[5]	Mwright	5, 3, 0, 0 ,0	[5]
Eigminc	22, 22, 0, 0, 0	[5]	Mhw4d	5, 3, 0, 0, 0	[5]
Ex5_2_4	7, 0, 1, 3, 2	[5]	Madsen	3, 0, 0, 6 ,0	[5]
Ex9_1_4	10, 4, 5, 0, 0	[5]	Minmaxrb	3, 0, 0, 2, 2	[5]
Ex8_4_2	24, 10, 0, 0, 0	[5]	Median_scop_vareps	5, 0, 0, 3, 0	[5]
Ex9_2_5	7, 3, 4, 0, 0, 0	[5]	Matrix2	6, 0, 0, 2, 0	[5]
Ex14_1_5	6, 0, 4, 2, 0	[5]	Mistake	9, 0, 0, 12, 0	[5]
Ex9_2_6	16, 6, 6, 0, 0	[5]	O32	5, 0, 0, 6, 0	[5]
Ex9_2_7	10, 4, 5, 0, 0	[5]	Pgon	12, 0, 0, 15, 5	[5]
Ex9_1_2	10, 4, 5, 0, 0	[5]	Robot	14, 2, 0, 0, 0	[5]
Ex2_1_9	10, 0, 1, 0, 0	[5]	Rk23	17, 7, 4, 0, 0	[5]
Ex2_1_3	13, 0, 0, 0, 9	[5]	S381	13, 0, 1, 0, 3	[27]
Ex8_4_1	22, 10, 0, 0, 0	[5]	S355	8, 5, 0, 0, 0	[27]
Ex8_4_5	15, 11, 0, 0, 0	[5]	S336	3, 1, 1, 0, 0	[27]
F_e	7, 0, 0, 3, 4	[11]	S262	4, 0, 1, 0, 3	[27]
Fermat_scop_vareps	5, 0, 0, 3, 0	[27]	S203	5, 3, 0, 0, 0	[27]
Fp_2_1	6, 0, 0, 1, 1	[11]	Springs_nonconvex	32, 0, 0, 10, 0	[27]
Genhs28	10, 0, 8, 0, 0	[5]	Steifold	4, 3, 0, 0, 0	[2]
Hs087	11, 4, 2, 0, 0	[5]	Sample	4, 0, 0, 2, 0	[27]
Hs108	9, 0, 0, 12, 0	[5]	Hs080	5, 3, 0, 0, 0	[5]

Dimension is denoted by D; NE, the number of nonlinear equations; LE, the number of linear equations; NIE, the nonlinear inequalities; and the number of linear inequalities is LIE

(iii) IP with box ranking according to maximum $\overline{F}(Y)$ augmented with penalty, $(\overline{F}(Y) - INF(Y)^2$, described in [37]); and IP with the double swapping criteria (max $INF(Y)/$ max $\overline{F}(Y)$) box ranking approach. We name the proposed double criteria approach also the no penalty approach.

The first set of IP variants listed above enable us to measure the impact of the branching rule, IIR against rules established in the interval literature. The second set of variants enable the comparison of the adaptive tree search management approach against classical tree management techniques. The third set of variants enable the comparison of the swapping double criteria box ranking method against the static penalty method. We run all three branching rules (Rule A, Rule C, and IIR) both with the augmented $\overline{F}(Y)$ box selection approach and the proposed swapping criterion approach. All these runs used the three tree management approaches (depth-first, best-first, and adaptive). However, the depth-first approach does not require box ranking by default, hence, we have 15 different IP combinations (runs) for each test problem.

4.2 Results

We measured the performance of each IP variant on all benchmarks in terms of the following performance indicators: the average absolute deviation from the global optimum (abbreviated in the tables as DfO) over all 55 plus 5 benchmarks obtained within the allowed CPU time limit, the average CPU time in STUs, the average number of tree stages where IP stops (NoS), the average number of times FSQP is invoked (NoL), the average number of function calls invoked outside FSQP (NoF), the number of best solutions obtained (NoB) and the number of problems where a feasible solution could not be obtained (number of unsolved problems within the CPU time limit, NoU). We provide two summaries of results: one for the 55 non-trigonometrical problems and one for the 5 trigonometric ones. The reason for this division is that BARON is not able to solve trigonometric models. The numerical results are provided in Tables 6 and 7 for non-trigonometric and trigonometric problems, respectively.

When we compare the three tree management schemes for IP in Table 6, we observe that the overall best deviations from the optimum are obtained by IIR-adaptive tree management scheme under the no penalty box ranking scheme. This observation is also confirmed by the fact that all three rules in this configuration have the lowest number of problems where IP did not converge to a feasible solution. The performance of the widest side rule (Rule A) is close to that of IIR under adaptive tree management scheme in the no penalty box ranking approach. Rule A performs best only under best-first/no penalty box ranking configuration. In other configurations Rule A is inferior to IIR. Rule C (maximum smear) is usually the worst performing rule under all configurations except for the depth-first approach where it is close to IIR. In summary, best results are obtained by the IIR—adaptive—no penalty and the Rule

Table 6 Summary of results obtained on 55 non-trigonometric COP benchmarks

Method		DfO	STU	NoS	NoL	NoF	NoB	NoU
IP adaptive, no penalty	IIR	0.55	0.317	4.84	1,502	28,703	44	0
	Rule A	0.61	0.341	5.82	1,311	27,542	43	0
	Rule C	8.44	0.564	4.17	3,132	28,146	33	6
IP adaptive, penalty	IIR	0.92	0.421	4.98	1,382	25,831	41	2
	Rule A	2.27	0.458	5.89	1,272	29,824	39	1
	Rule C	7.70	0.607	4.32	2,498	27,896	35	5
IP best first, no penalty	IIR	1.83	1.656		861	19,402	39	3
	Rule A	0.58	1.791		894	22,604	41	2
	Rule C	3.87	2.467		985	11,589	30	12
IP best first, penalty	IIR	0.79	1.847		854	16,654	38	5
	Rule A	2.27	1.928		1,056	19,678	40	1
	Rule C	7.31	2.827		982	11,709	31	9
IP depth first	IIR	0.79	1.847		854	16,654	38	5
	Rule A	2.27	1.928		1,056	19,678	40	1
	Rule C	7.31	2.827		982	11,709	31	9
FSQP		5.54	0.000				31	13
Baron		0.43	0.067				45	0
Conopt		14.54	0.000				35	9
LGO		1.90	0.079				40	8
Minos		14.66	0.000				35	9
Snopt		16.36	0.000				34	9

The following abbreviations were used: DfO, average deviation from optimum; STU, average CPU time needed in STU; NoS, average number of stages; NoL, average number of FSQP calls; NoF, average number of objective function calls; NoB, number of best solutions; NoU, number of unsolved problems

Table 7 Summary of results obtained on 5 trigonometric COP benchmarks

Method		DfO	STU	NoS	NoL	NoF	NoB	NoU
IP adaptive, no penalty	IIR	0.02	0.185	4.00	1, 233	27, 610	4	0
	Rule A	0.03	0.286	3.60	1, 476	27, 718	4	0
	Rule C	0.00	0.295	3.40	2, 126	27, 684	4	1
IP adaptive, penalty	IIR	0.02	0.251	5.40	1, 214	933	4	0
	Rule A	0.03	0.269	3.80	1, 330	27, 718	4	0
	Rule C	0.00	0.284	3.60	1, 818	27, 684	4	1
IP best first, no penalty	IIR	0.05	1.739		880	22, 458	4	0
	Rule A	0.03	1.781		800	14, 044	4	0
	Rule C	0.00	1.311		876	13, 490	4	1
IP best first, penalty	IIR	0.11	1.711		863	21, 506	4	0
	Rule A	0.04	1.708		951	16, 173	4	0
	Rule C	0.00	2.268		919	10, 337	4	1
IP depth first	IIR	0.16	0.041		387	27, 627	4	0
	Rule A	0.10	0.032		401	27, 682	4	0
	Rule C	0.00	0.128		1, 322	27, 718	4	1
FSQP		1.17	0.000				2	1
Conopt		40.78	0.000				1	0
LGO		2.23	0.003				1	1
Minos		104.14	0.000				0	0
Snopt		38.08	0.000				1	0

The following abbreviations were used: DfO, average deviation from optimum; STU, average CPU time needed in STU; NoS, average number of stages; NoL, average number of FSQP calls; NoF, average number of objective function calls; NoB, number of best solutions; NoU, number of unsolved problems

A—best-first—no penalty combinations. However, in terms of the number of best solutions obtained, and the CPU time, IIR—adaptive—no penalty is better than its competitors.

An advantage of the adaptive tree management scheme is that it reduces CPU times due to relieved memory requirements and reduced computation times due to a less number of box sorting operations as compared to the best-first approach. Furthermore, it is effective in sending the correct boxes (intervals that contain feasible solutions) to FSQP that converges to feasible solutions in a less number of iterations as compared to other tree management schemes. As expected, the best-first approach is the slowest one among all three tree management schemes (due to maintaining lengthy sorted box lists) and the fastest one is the depth-first.

However, the fastest approach is significantly inferior in solution quality in terms of unsolved problems, number of best solutions found and average absolute deviation from the optimum. For the depth-first approach, the performance of Rule C and IIR are not significantly different and that of Rule A is quite inferior. A final observation is that the no penalty box ranking method is generally better than the penalty one in both best-first and adaptive tree management approaches with respect to the sum of all three partitioning rules' average deviations from the global optimum.

When we compare IP solvers to others, we can observe that the two complete solvers BARON and LGO are best performing. The performance of BARON is better than the best IP configuration (IIR/adaptive/no penalty) both in terms of average deviation from the optimum and CPU time. LGO's performance is somewhat inferior to that of BARON in non-trigonometric problems. The third best non-IP solver, the stand-alone FSQP, is much worse than LGO. The difference in performance between FSQP and the best IP configuration that uses FSQP as a local solver illustrates the strength of it called by a complete solver.

In the results of the trigonometric problems provided in Table 7 show that the relative performance of different IP configurations is quite similar to our findings in Table 6. The zero deviation of Rule C is due to its incapability of solving one problem whereas the other two rules converge in all test instances. Hence, the given average deviations of other rules are due to a single problem. Solvers other than IP are significantly inferior on these test problems as compared to any IP configuration and they do not provide the best solution in more than 2 problems out of five.

Acknowledgements The authors wish to thank Andre Tits (University of Maryland) for providing the source code of CFSQP, and Hermann Schichl (Universität Wien) for his valuable comments and suggestions, which improved the paper.

References

1. Alefeld, G., Herzberger, J.: Introduction to Interval Computations. Academic Press, New York (1983)
2. Balogh, J., Tóth, B.: Global optimization on Stiefel manifolds: a computational approach. CEJOR **13**, 213–232 (2005)
3. Benhamou, F., McAllester, D., Van Hentenryck, P.: CLP(intervals) revisited. In: Proc. of ILPS'94, pp. 124–138 (1994)
4. Casado, L.G., García, I., Csendes, T.: A new multisection technique in interval methods for global optimization. Computing **65**, 263–269 (2000)
5. COCONUT, http://www.mat.univie.ac.at/~neum/glopt/coconut/
6. Csendes, T., Rapcsák, T.: Nonlinear coordinate transformations for unconstrained optimization. I. Basic transformations. J. Global Optim. **3**, 213–221 (1993)
7. Csendes, T., Ratz, D.: A review of subdivision direction selection in interval methods for global optimization. ZAMM **76**, 319–322 (1996)
8. Csendes, T., Ratz, D.: Subdivision direction selection in interval methods for global optimization. SIAM J. Numer. Anal. **34**, 922–938 (1997)
9. Dallwig, S., Neumaier, A., Schichl, H. : GLOPT—a program for constrained global optimization. In: Bomze, I.M., Csendes, T., Horst, R., Pardalos, P.M. (eds.) Developments in global optimization, pp. 19–36. Kluwer, Dordrecht (1997)
10. Drud, A.S.: CONOPT: A System for Large Scale Nonlinear Optimization. Reference Manual for CONOPT Subroutine Library, ARKI Consulting and Development A/S, Bagsvaerd, Denmark (1996)
11. Epperly, T.G.: Global optimization of nonconvex nonlinear programs using parallel branch and bound. PhD dissertation, University of Wisconsin-Madison, USA (1995)
12. Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: an SQP algorithm for large-scale constrained optimization. Numerical Analysis Report 97-2, Department of Mathematics, University of California, San Diego, La Jolla, CA (1997)
13. Hansen, E.R.: Global Optimization Using Interval Analysis. Marcel Dekker, New York (1992)
14. Hansen, E., Sengupta, S.: Global constrained optimization using interval analysis. In: Nickel, K.L. (ed.) Interval Mathematics, Academic Press, New York (1980)
15. Kearfott, R.B.: Decompostion of arithmetic expressions to improve the behaviour of interval iteration for nonlinear systems. Computing **47**, 169–191 (1991)
16. Kearfott, R.B.: An overview of the GlobSol Package for Verified Global Optimization. Talk given for the Department of Computing and Software. McMaster University, Ontario, Canada (2003)
17. Kearfott, R.B., Manuel, N. III.: A portable interval Newton/bisection package. ACM Trans. Math. Software **16**, 152–157 (1990)
18. Knüppel, O.: PROFIL/BIAS—a fast interval library. Computing **53**, 277–287 (1994)
19. Korf, R.E.: Depth-first iterative deepening: An optimal admissible tree search. Artif. Intell. **27**, 97–109 (1985)
20. Markót, M.C.: Reliable global optimization methods for constrained problems and their application for solving circle packing problems. PhD dissertation, University of Szeged, Hungary (2003)
21. Markót, M.C., Fernandez, J., Casado, L.G., Csendes, T.: New interval methods for constrained global optimization. Math. Program. **106**, 278–318 (2006)
22. Murtagh, B.A., Saunders, M.A.: MINOS 5.0 User's Guide. Report SOL 83-20, Department of Operations Research, Stanford University, USA (1987)
23. Pedamallu, C.S., Özdamar, L., Csendes, T.: An interval partitioning approach for continuous constrained optimization. In: Models and Algorithms in Global Optimization, pp. 73–96. Springer, Berlin (2006)

24. Pedamallu, C.S., Özdamar, L., Csendes, T.: Symbolic interval inference approach for subdivision direction selection in interval partitioning algorithms. J. Global Optim. **37**, 177–194 (2007)
25. Pedamallu, C.S., Pósfai, J., Csendes, T.: Interval partitioning algorithm for constraint satisfaction problems. Int. J. of Model. Identif. Control. (accepted for publication)
26. Pintér, J.D.: LGO—a program system for continuous and Lipschitz global optimization. In: Bomze, I.M., Csendes, T., Horst, R., Pardalos, P.M. (eds.) Developments in Global Optimization, pp. 183–197. Kluwer, Boston (1997)
27. PrincetonLib.: Princeton Library of Nonlinear Programming Models. http://www.gamsworld.org/performance/princetonlib/princetonlib.htm
28. Rapcsák, T., Csendes, T.: Nonlinear coordinate transformations for unconstrained optimization. II. Theoretical background. J. Global Optim. **3**, 359–375 (1993)
29. Ratschek, H., Rokne, J.: New computer Methods for Global Optimization. Ellis Horwood, Chichester (1988)
30. Ratz, D., Csendes, T.: On the selection of subdivision directions in interval branch-and-bound methods for global optimization. J. Global Optim. **7**, 183–207 (1995)
31. Robinson, S.M.: Computable error bounds for nonlinear programming. Math. Program. **5**, 235–242 (1973)
32. Sahinidis, N.V.: Global optimization and constraint satisfaction: the branch-and-reduce approach. In: Bliek, C., Jermann, C., Neumaier, A. (eds.): COCOS 2002, LNCS, pp. 1–16 vol. **2861** (2003)
33. Shcherbina, O., Neumaier, A., Sam-Haroud, D., Vu, X.-H., Nguyen, T.-V.: Benchmarking global optimization and constraint satisfaction codes. LNCS **2861**, 211–222 (2003)
34. Smith, E.M.B., Pantelides, C.C.: A symbolic reformulation/spatial branch and bound algorithm for the global optimization of nonconvex MINLP's. Comp. Chem. Eng. **23**, 457–478 (1999)
35. Tawarmalani, M., Sahinidis, N.V.: Global optimization of mixed-integer nonlinear programs: a theoretical and computational study. Math. Program. **99**, 563–591 (2004)
36. Wolfe, M.A.: An interval algorithm for constrained global optimization. J. Comput. Appl. Math. **50**, 605–612 (1994)
37. Yeniay, O.: Penalty function methods for constrained optimization with genetic algorithms. Math. Comp. Appl. **10**, 45–56 (2004)
38. Zhou, J.L., Tits, A.L.: An SQP algorithm for finely discretized continuous minimax problems and other minimax problems with many objective functions. SIAM J. Optim. **6**, 461–487 (1996)