J Glob Optim (2007) 37:177–194
DOI 10.1007/s10898-006-9043-y

ORIGINAL PAPER

Symbolic interval inference approach for subdivision direction selection in interval partitioning algorithms

Chandra Sekhar Pedamallu · Linet Özdamar · Tibor Csendes

Received: 22 August 2005 / Accepted: 5 March 2006 /
Published online: 29 June 2006

Abstract In bound constrained global optimization problems, partitioning methods utilizing Interval Arithmetic are powerful techniques that produce reliable results. Subdivision direction selection is a major component of partitioning algorithms and it plays an important role in convergence speed. Here, we propose a new subdivision direction selection scheme that uses symbolic computing in interpreting interval arithmetic operations. We call this approach symbolic interval inference approach (SIIA). SIIA targets the reduction of interval bounds of pending boxes directly by identifying the major impact variables and re-partitioning them in the next iteration. This approach speeds up the interval partitioning algorithm (IPA) because it targets the pending status of sibling boxes produced. The proposed SIIA enables multi-section of two major impact variables at a time. The efficiency of SIIA is illustrated on well-known bound constrained test functions and compared with established subdivision direction selection methods from the literature.

Keywords Box-constrained global optimization · Interval branch and bound methods · Symbolic computing · Subdivision direction selection

1 Introduction

Interval partitioning algorithms (IPA) use interval arithmetic (see e.g. Moore 1966) to produce reliable results for constrained and unconstrained optimization (for an

C. S. Pedamallu
School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore
e-mail: chandra@inf.u-szeged.hu; pcs_murali@lycos.com

L. Özdamar
Izmir Ekonomi Universitesi, Izmir, Turkey
e-mail: linetozdamar@lycos.com

T. Csendes(✉)
University of Szeged, Institute of Informatics, Szeged, P.O. Box 652, H-6701, Hungary
e-mail: csendes@inf.u-szeged.hu

overview, see Hansen 1992; Ratschek and Rokne 1995). Due to their reliability, interval applications take place in a wide scientific field (Kearfott and Kreinovich 1996). In bound constrained global optimization problems, IPA subdivides the given domain into smaller subspaces (boxes) that are assessed according to the function range calculated by using an approximating inclusion function. Based on the function range bounds and a known best solution that is updated during the search, some subspaces are deleted reliably, because they cannot hold the global optimum solution (Hammer et al. 1993; Pintér 1992). Subdivision continues in remaining boxes so that the location of the global optimum solution can be enclosed within a small box of a given tolerance. The final report contains all such boxes in the given function domain.

Convergence rate of IPA depends on the use of accelerating devices (such as monotonicity and concavity tests) that help in discarding boxes (Ratschek and Rokne 1988, 1995) and on the selection of subdivision direction (variable whose domain is to be re-partitioned) (Berner 1996; Csendes and Ratz 1996; Csendes and Ratz 1997; Csendes et al. 2000; Hansen 1992; Moore 1966; Neumaier 1990; Ratz and Csendes 1995). In IPA, the latter issue has a major impact on convergence rate because reducing the domain size of a specific variable might enhance the reduction in the overestimated function range of the sibling boxes to a significant degree. Thereby, boxes that cannot be discarded due to their promising overestimated upper bounds may become disposable in a few re-partitioning iterations with a good subdivision direction selection strategy.

Subdivision rules proposed up to date are based on criteria such as the width of variable intervals, or estimated function improvement by selected variables (gradient information). The performance of such rules is assessed extensively on standard test problems (Csendes and Ratz 1996, 1997; Csendes et al. 2000; Ratz and Csendes 1995) resulting in the general conclusion that gradient based rules work much better.

In Berner (1996), these rules are converted into parallel multi-section rules by taking the first k number of variables from a list of variables sorted according to the rule (called k-best strategy here). Multi-section (subdivision of some variables in parallel) and multi-splitting (subdivision of a single variable's width into $s > 2$ pieces) approaches are proposed in Csallner et al. (2000a, b). The latter studies investigate the efficiency related to specific values of s with regard to each subdivision rule. Casado et al. (2001) propose multi-section/multi-splitting hybrids by subdividing intervals of all variables into 2 or more pieces (s^n) in parallel. The authors propose a parametric method that involves the comparison of a box assessment criterion with given constants used in deciding which hybrid parallel scheme should be used for a given box. In Casado et al. (2001) the authors use the box assessment criterion as a box selection rule and utilize multi-section subdivision rules based on k-best strategy found in Berner (1996).

Here, we propose a symbolic computing—interval partitioning cooperation scheme for enhancing the process of subdivision direction selection. In the literature, symbolic-interval cooperation frameworks are proposed mostly for solving constraint satisfaction problems (Ceberio and Granvilliers 2000; Granvilliers et al. 2001; Granvilliers 2004; Lhomme et al. 1998; Sam-Haroud and Faltings 1996). In particular, consistency techniques (Sam-Haroud and Faltings 1996) and interval propagation through multiple constraints are proposed to reduce variable domains so that feasible regions can be identified (see hull and box consistency techniques (Granvilliers et al. 2001; Sam-Haroud and Faltings 1996). Here however, symbolic-interval cooperation is developed to propagate intervals through different subexpression complexity levels

of a function. While past symbolic-interval cooperation was based on the full function expression, the proposed cooperation propagates intervals at hierarchically recursive subexpression levels. The propagation is exhaustive and it identifies a couple of major impact variables (source variables) that provide exactly the relevant bound of the function's interval over a given box (in unconstrained maximization, this bound is the upper bound of function range). We call this identification procedure symbolic interval inference approach (SIIA). The subdivision direction selection rule developed from SIIA is called symbolic inference rule (SIR). SIR's goal is to reduce the domain of the source variables with a guarantee of narrowing down function range overestimation in sibling boxes.

In this framework, SIR is integrated with IPA and it is activated at every box assessment during execution. Here, to enable such a symbolic propagation, we develop three basic components: a parser, a tree builder, and a rule operator. The tree builder constructs a binary tree that represents a given function after parsing. The rule operator uses the binary tree for propagating intervals at the subexpression levels in order to make an inference on the source variables. Source variables are subdivided in parallel in the next iteration. Hence, the proposed method also includes a multi-section method that subdivides 2 variables at a time (an exception occurs when all variables but one have too small interval widths to be subdivided). In our implementation, source variable intervals are bisected in sibling boxes, however, multi-splitting can be applied easily depending on the specific impact of each source variable. In the following sections, the essential components of SIIA, the convergence property of SIR and its implementation in IPA are described. Then, numerical experiments are conducted on well-known test problems from the literature in order to assess the performance of SIR against k-best (for a fair comparison, 2-best) parallel version of established subdivision direction selection rules and against the standard 2^n multi-section rule. It is shown that SIR is effective in improving the convergence rate of IPA.

2 Interval partitioning algorithms: Proposed convergence criterion

2.1 Basics of IPA and terminology

Bound constrained global optimization problems are expressed as:

$$\max f(x) \tag{2.1}$$

where $X \subseteq \mathbb{R}^n$ is the search box and f: $X \to \mathbb{R}$, is the objective function. The search box is assumed to be a closed interval and it is denoted as $[\underline{X}, \overline{X}]$, where $\underline{X}_j = \min X_j$ and $\overline{X}_j = \max X_j$, for $j = 1, 2, \ldots, n$. A global maximizer is denoted as x^*. Denote the set of compact intervals by $\mathbb{I} := \{[a, b] \mid a \leq b;\ a, b \in \mathbb{R}\}$ and the set of n-dimensional intervals (also called intervals or boxes) by $\mathbb{I}^n$. The *width* of an interval X is defined by $w(X) = \overline{X} - \underline{X}$. The definition of an inclusion function and its fundamental properties are provided below.

Definition 2.1 Let $f(Y) = \{f(x) : x \in Y\}$ be the range of f over $Y \in \mathbb{I}(X)$, where $\mathbb{I}$ is the set of n-dimensional compact intervals in X. A function $F : \mathbb{I}(X) \to \mathbb{I}$ is an *inclusion function* for f, if $f(Y) \in F(Y)$ for any $Y \in \mathbb{I}(X)$.

Definition 2.2 An interval function F is said to be *inclusion isotone* if for any pair of boxes Y and $Z \subseteq \mathbb{I}(X)$, $Y \subseteq Z$ implies $F(Y) \subseteq F(Z)$.

It is assumed that for the studied functions the natural interval extension of f over Y is always defined in the real domain. Furthermore, F is α-*convergent* over X, that is, for all $Y \in \mathbb{I}(X)$, $w(F(Y)) - w(f(Y)) \leq cw(Y)^{\alpha}$ where c and α are positive constants.

IPA subdivides X into smaller boxes that are assessed with respect to their potential of holding a global optimal solution. Basically, IPA is categorized as a Branch and Bound technique in the real domain. The following section summarizes box assessment.

2.2 Optimality status of boxes and convergence criterion

In a partitioning algorithm, each box Y is assessed for its optimality status by calculating the bounds of $F(Y)$ with an *Interval Library* such as PROFIL (Knüppel 1994). The concepts related to a box's optimality status are discussed below.

Suppose that the objective function value of a known solution is available as a current lower bound (CLB) for $f(x)$. Boxes are classified according to the following rules.

Definition 2.3 (Cut-off test) If $\overline{F}(Y) < \text{CLB}$, then the box Y is called a *suboptimal box* and it is deleted because it cannot contain x^*.

Definition 2.4 If $\underline{F}(Y) \leq \text{CLB}$ and $\overline{F}(Y) > \text{CLB}$, then the box Y is called a *pending box*. A pending box holds the potential of containing x^*.

Definition 2.5 The pending status or potential of a pending box is defined as:

$$P_Y = \overline{F}(Y) - \text{CLB}. \tag{2.2}$$

When a box is pending, more advanced optimality tests (accelerating devices) such as monotonocity, and nonconvexity test can be applied to discard it (Jansson and Knüppel 1995; Ratschek and Rokne 1988, 1995).

In each box assessment, the function range estimate $F(M)$ over a sufficiently small box M enclosing the mid-point of Y is calculated. In the assessment of the first box, $\min f(M)$ becomes the current lower bound (CLB) and each time a better mid-point solution is found, CLB is updated.

IPA continues to subdivide available pending boxes until either they are all deleted or interval sizes of all variables in existing boxes are less than a given tolerance, δ. All such boxes are reported as potential boxes that may contain x^*. In Fig. 1, a generic pseudocode is provided for IPA.

In essence, IPA aims to discard suboptimal boxes and reduce the number of pending boxes with as few function calls as possible. This is facilitated by partitioning appropriate variables and generating subboxes whose overestimation in P_Y is reduced. Then, the algorithm converges fast by discarding suboptimal boxes early and also by partitioning promising boxes in a fitting direction to reach the global basin of attraction. While variable selection is made according to this criterion, box selection is carried out following a worst-first strategy, i.e. the box with the maximum P_Y is selected first. We would like to mention that P_Y is a traditional box selection index used in IPA. A normalized version of this index (the *RejectIndex*) is obtained by dividing P_Y by $w(F(Y))$ (Casado et al. 2001). The *RejectIndex* aims at reducing the overestimation in smaller boxes with greater uncertainty whereas we target at discarding large boxes. Below, we define a convergence criterion based on the pending status of boxes and show that IPA is convergent with respect to the latter. We assume for the whole study that the

Notation : *WLB*: Working List of Boxes; M : Point interval at the mid-point of a box;
$F(M)$: range estimate at M ; δ: tolerance for final interval length

```
Void IPA:
{
 Construct tree structure for f(x);
 Initialize: initial box =I(X); CLB = -∞; WLB = I(X);
 While WLB ≠ φ do
 {
  Select a box Y∈ WLB; Calculate F(Y);
  if ( F̄(Y) > CLB) AND (At least for one variable interval, w(x_i) > δ )
   {
    if ( F(Y) > CLB), then CLB = F(Y) ;
    Calculate the mid-point function value, F(M);
    if (F(M) > CLB), then CLB = F(M);
    Select subdivision direction;     // Activate Symbolic Interval Inference Rule;
    Subdivide Y to obtain four sibling boxes: S1, S2, S3, S4; // Multisection - 4 siblings
    WLB = WLB – {Y}; WLB = WLB + {S1, S2, S3, S4};
   } // endif
  else
   {
    if (w(x_i) < δ , ∀i), then store Y; WLB = WLB – {Y};
   }
 } // endwhile
 Report all stored boxes;
} // endprocedure
```

Fig. 1 Generic pseudocode for IPA

subdivision direction selection is balanced, i.e. each coordinate direction appears in the sequence of subdivision an infinite number of times (Csendes and Ratz 1997).

Lemma 2.6 *IPA reduces the pending status of boxes by nested partitioning where the widths of the subdivided boxes tend to zero.*

Proof Consider a *pending* box Y. Suppose a variable is re-partitioned to result in two sibling boxes V and W. By the isotone inclusion property of F, the following holds for V and W:

$$\overline{F}(Y) \geq \overline{F}(V) \text{ and } \overline{F}(Y) \geq \overline{F}(W). \tag{2.3}$$

In the worst case, even if CLB does not improve in sibling boxes, i.e., $\text{CLB}^V = \text{CLB}^Y$, since P_Y is a function of $\overline{F}(Y)$, and

$$P_Y - P_V \geq 0. \tag{2.4}$$

Hence, the reduction in the pending status of siblings is always non-negative, and given a box Y_j that contains x^*, the pending status goes to zero in the limit as the number of nested re-partitioning iterations j grows, (utilizing the α-*convergence*). That is,

$$\lim_{j\to\infty} \overline{F}(Y_j) \to \text{CLB}. \tag{2.5}$$

While boxes that do not contain x^* are discarded by the cutoff test due to the reduction in their pending status, the optimal box has $\overline{F}(Y_j) \to f(x^*)$ in the limit. □

Convergence properties of subdivision rules proposed in the literature are generally based on a balanced bisection, e.g. on bisection along the largest width interval variable. Convergence of those rules are guaranteed in the sense that in the limit, as re-partitioning iterations increase, a sufficiently fine partition provides an enclosure for the global optimum (Ratschek and Rokne 1988). Some rules based on gradient information require the application of monotonicity test in IPA to guarantee convergence (Ratz and Csendes 1995). The proposed criterion only uses the property of inclusion isotonicity, the α-convergence, and it does not require any additional assumptions.

3 Symbolic interval inference approach (SIIA) for subdivision direction selection

The proposed SIIA has three enabling components: a parser, a tree builder, and a rule operator. The parser is activated once before IPA is executed. It dissects the function expression and passes the output to the tree builder. A binary tree that represents the function with all its subexpressions is then constructed. The contribution of subexpressions and atomic elements (variables) to the function range are recursively calculated by calling an *Interval Library* at each (molecular) level of the hierarchical binary tree so that the impact of all terms can be assessed in descending order of complexity.

At each box assessment, SIR activates a tree traversal or labeling procedure to identify the pair of variables to be re-partitioned. Since P_Y is a function of $\overline{F}(Y)$, SIR labels $\overline{F}(Y)$ to reduce P_Y at the root node (function expression). Then, SIR labels the interval bound resulting in the label value at the root node and goes down the tree until the first atomic element (variable) having the maximum impact on $\overline{F}(Y)$ is reached. Then, a backward traversal is activated to identify the coupling maximum impact (source) variable. This couple is re-partitioned in the next iteration to form 2^2 siblings in parallel. A second variant of SIR is obtained by selecting the subexpression with the largest interval width rather than the maximum bound one. In case of ties among subexpression nodes, the one with the maximum bound can be chosen. Both variants of SIR have been tested in this paper.

3.1 The tree builder: Binary tree representation

Binary tree representation of expressions enables the execution of SIR. 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 $\circ$ Right). $\circ$ can be a binary arithmetic operator $(*, +, -, /)$ having two branches ("Left", "Right") or a unary mathematical function such as $\ln, \exp, \sin$, etc. having the argument of the function always placed in the "Left" branch. We provide the following expression (Eq. 3.1) as an example to be used throughout this paper for illustrating the mechanics of SIIA's three components:

$$((x_1 + x_2) * (x_3 + x_4)) + \sin(x_1 + x_3). \tag{3.6}$$

In Eq. 3.1, the partial expression "$\sin(x_1 + x_3)$" contains one unary operator (sine) that always branches out to its left, however, the addition operator within the sine operator is a binary operator connecting x_1 and x_3. The binary tree pertaining to this example is illustrated in Fig. 2.

3.2 Rule operator: Interval propagation through a binary tree

Interval bounds for subexpressions (intermediate nodes) are calculated with a bottom-up tree traversal. First, the interval ranges of each leaf (variable or constant) are substituted into the subexpressions at the next higher level by using the connecting operators. This process is repeated by accessing the next higher level until the root node is reached. The pseudocode of the rule is given in Fig. 3 and propagated intervals for the expression in (3.1) are illustrated in Fig. 2.

This recursive propagation is realized using the monotonicity property of elementary interval operations (binary operator) and functions (unary operator). Given the fact that Q, G, and H are isotone inclusion functions, for any recursive definition of arithmetical expression $q = h \circ g$, the range $q(Y)$ is accurately represented by $Q(Y) = G(Y) \circ H(Y)$. Consequently, interval propagation over a binary tree results in an accurate calculation of subexpression intervals.

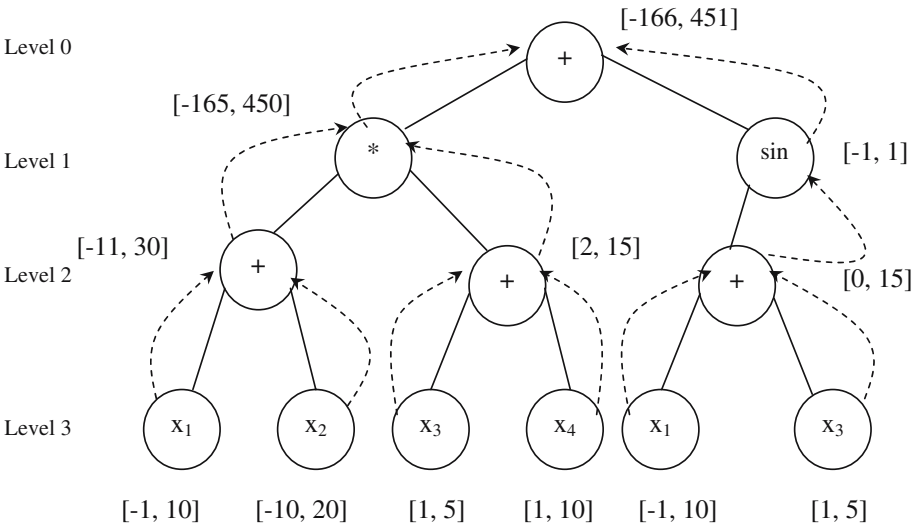

Fig. 2 Interval propagation for "$((x_1 + x_2) * (x_3 + x_4)) + \sin(x_1 + x_3)$"

Node_Type *SIR (Node_Type Node)* {
if (node_level $k = 0$), $bnd = \overline{F}(Y)$;
else $bnd = \Lambda^{k}$;
Identify the pair a Θ b ∈
$\{\{\underline{L}^{k-1}\Theta\underline{R}^{k-1}\}, \{\underline{L}^{k+1}\Theta\overline{R}^{k+1}\}, \{\overline{L}^{k+1}\Theta\underline{R}^{k+1}\}, \{\overline{L}^{k+1}\Theta\overline{R}^{k+1}\}\} : a\ \Theta b = bnd$;
$\Lambda^{k+1} = MAX\ \{|a|, |b|\}$;
if $\Lambda^{k+1} = |a|$, then return the Left branch node as labeled at level $k+1$;
else return the Right branch node as labeled at level $k+1$;
}

Fig. 3 Pseudocode for SIR-*bounds* (Input: node at level k; Output: labeled node at level $k + 1$)

3.3 Symbolic Inference Rule (SIR) and Labeling Procedure SIR_Tree

In the maximum bound variant of SIR (SIR-*bounds*), one interval bound is labeled at a time at each level of the tree by executing forward and backward chaining to end up with the pair of source variables (leaves) that contribute most to $\overline{F}(Y)$. The couple of source variables identified are subdivided in the next iteration.

Suppose we proceed to identify a source variable on the binary tree of a function, starting from the root node. There are two possible branches to take from any parent node. From here on, we denote a parent at tree level k as D^k, and the nodes Left and Right that are its subbranches, as L^{k+1} and R^{k+1}. Further, we define Λ^k as labeled bound at level k.

Let us also denote the interval bounds of parent node D^k by $[\underline{D}^k, \overline{D}^k]$, and those of the subbranches as $[\underline{L}^{k+1}, \overline{L}^{k+1}]$ and $[\underline{R}^{k+1}, \overline{R}^{k+1}]$. As mentioned before, we label $\overline{D}^0$, i.e., $\overline{F}(Y)$, at root level (level zero) of the tree so as to reduce P_Y and result in a convergent rule.

For the root node, we determine which pair of interval bounds ($\{\underline{L}^1 \circ \underline{R}^1\}, \{\underline{L}^1 \circ \overline{R}^1\}$, $\{\overline{L}^1 \circ \underline{R}^1\}, \{\overline{L}^1 \circ \overline{R}^1\}$) results exactly in $\overline{D}^0$ when connected by their operator. Then, we compare the absolute values of individual bounds in the pair and take their maximum to choose the corresponding L or R branch. For instance, if $\{\underline{L}^1 \circ \underline{R}^1\} = \overline{D}^0$, and when $|\underline{L}^1| = \max\{|\underline{L}^1|, |\underline{R}^1|\}$, then we take the Left branch and label $|\underline{L}^1|$ to go down to the next level (level 2). This procedure is recursively applied from top to bottom, each time searching for the bound pair resulting in the labeled bound at the upper level till a leaf is hit. (Note that when a leaf is a constant, its counterpart is always selected, that is, a pair of subbranches that include a constant is treated as a unary operator.).

Once this forward tree traversal is over, all leaves in the tree corresponding to the variable selected are set to "Closed" status. The procedure then backtracks to the next higher level of the tree to identify the other leaf in the couple of variables that produce the labeled bound. Backtracking ends when the first "Open" leaf is encountered in this search. Hence, the couple of variables that contribute most to P_Y are identified. A formal procedure of SIR-*bounds* is given in Fig. 3. The pseudocode of the labeling algorithm, *SIR_Tree,* is given in Fig. 4. The start node is initialized as the root node.

```
Node_Type SIR _Tree (Node_Type Start_Node) {
 If ((Count > 2) OR (All leaves are "Closed")) then exit;
 Select_Node = SIR (Start_Node);     // calls procedure SIR
 If (Select_Node. Status = "Open Node")
    Start_Node = SIR_Tree(Select_Node);
 Else if (Select_Node. Status = "Open Leaf")  // found a source variable
    {
      Store source variable "Open Leaf";
       Close all leaves of type "Open Leaf";
      Count++;
      Start_Node = SIR_Tree (Next_Up(Select_Node));   // backtrack to identify second source
     }
    Else Start_Node = SIR_Tree (Next_Up(Select_Node)); // backtrack to identify second source
 Return Start_Node;
 }
```

Fig. 4 Procedure *SIR_Tree*:Recursive tree traversal of SIR. (Input: Root node; Output: pair of source leaves—variables)

Before procedure *SIR_Tree* is called for any box Y, all variables that have reached their positive tolerance widths (relative to the largest width of the variables that are used in the computation on the pending list) and that cannot be subdivided in the next iteration are set to "Closed" status. This is necessary, since otherwise the direction selection rule could choose only some of the possible subdivision directions, and that may endanger the convergence of the IPA.

As an alternative to the above described rule, SIR-*bounds*, we have also investigated another one (called SIR-*widths*), that chooses that branch of the computation tree which has the largest width of the expression inclusion related to the given node. In case the two widths are equal, we follow the branch according to SIR-*bounds*.

3.4 An illustration of SIR and SIR_Tree procedures

Suppose we have the example given in Fig. 2 with the expression interval $[-166, 451]$. Then, "451" is selected as the labeled bound Λ^0 at the root node. In SIR-*bounds*, we next determine which pair of interval bounds $(\{\underline{L}^1+\underline{R}^1\}, \{\underline{L}^1+\overline{R}^1\}, \{\overline{L}^1+\underline{R}^1\}, \{\overline{L}^1+\overline{R}^1\})$ results exactly in $\overline{D}^0$. The pair of interval bounds that provides 451 is (450, 1) since "450+1= 451". Hence, $\overline{L}^1 \circ \overline{R}^1 = \overline{D}^0$. We then compare the absolute values of individual bounds in this pair and take their maximum as the label at level $k+1$. $\Lambda^{k+1} = \max\{\overline{L}^1, \overline{R}^1\} = \overline{L}^1 = 450$. All steps of *SIR_Tree* for SIR-*bounds* and *SIR-widths* are provided below in detail and decisions are illustrated in Figs. 5 and 6 with bold arrows respectively.

In case of SIR-*bounds*, this leads to $\overline{R}^3$, a bound of leaf x_2. The leaf pertaining to x_2 is "Closed" from here onwards, and the procedure backtracks to Level 2. Then, SIR-*bounds* leads to the second source variable, x_1.

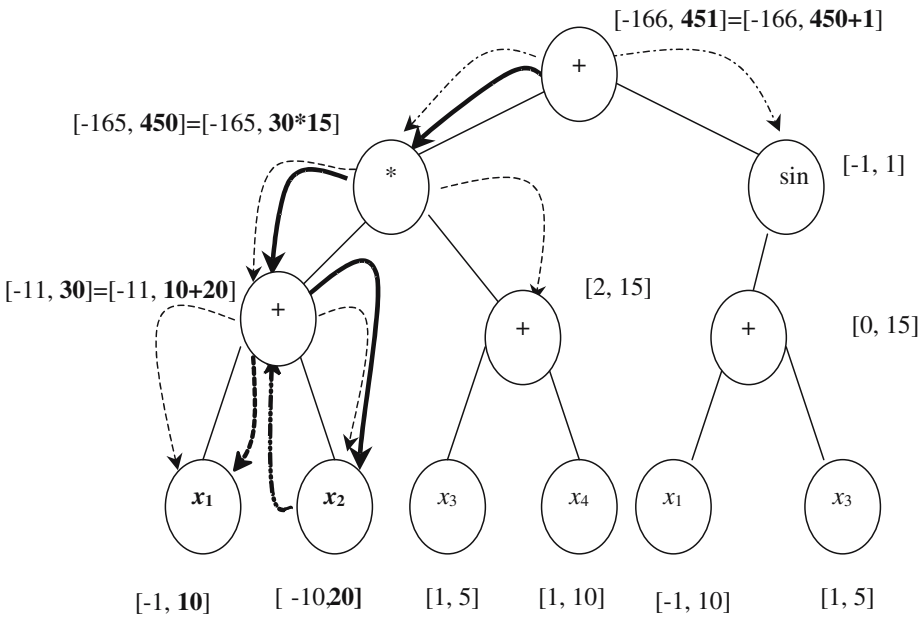

Fig. 5 Demonstration of the run of SIR-*bounds* on the example

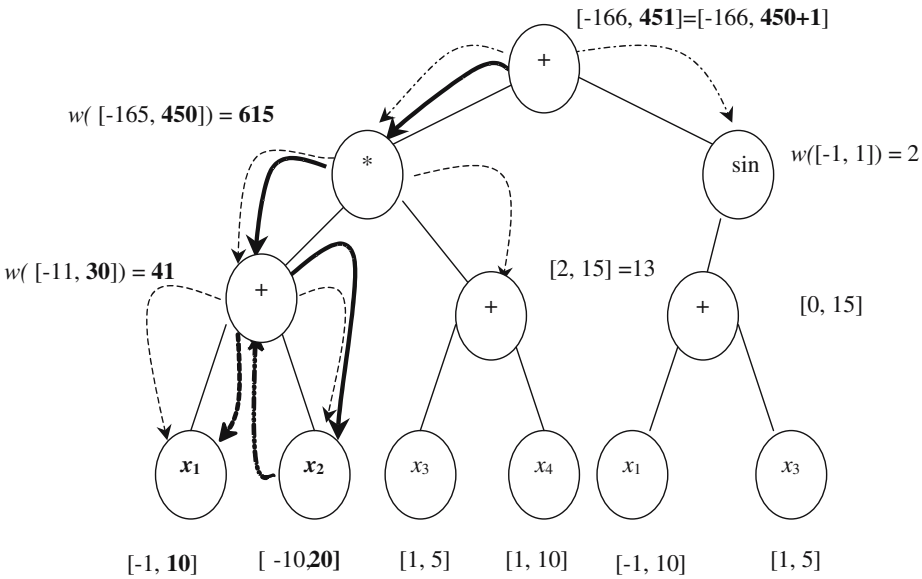

Fig. 6 Demonstration of the run of SIR-*bounds* on the example

In case of SIR-*widths*, this leads to L^3, a bound of leaf x_1. Then, the procedure backtracks to Level 2 and SIR-*widths* leads to the second source variable, x_2.

SIR-*bounds*	**SIR-*widths***
Level 0: $[\underline{D}^0, \overline{D}^0] = [-166, \mathbf{451}]$ $\Lambda^0 = \overline{D}^0$. $a \circ b = \{(-165+1)$ or $(450+1)$ or $(-165-1)$ or $(450-1)\}$ $= \mathbf{451}$. Hence, $a \circ b = \overline{L}^1 + \overline{R}^1$, and $\Lambda^1 = \max\{\lvert\overline{L}^1\rvert, \lvert\overline{R}^1\rvert\} = \max\{\lvert 450\rvert, \lvert 1\rvert\}$ $= \mathbf{450} = \overline{L}^1$. **Level 1:** $[\underline{D}^1, \overline{D}^1] = [-165, \mathbf{450}]$ $a \circ b = \{(-11 * 2)$ or $(30*2)$ or $(-11 * 15)$ or $(30 * 15)\}$ $= \mathbf{450} \Rightarrow a \circ b = \overline{L}^2 * \overline{R}^2$, $\Lambda^2 = \max\{\lvert\overline{L}^2\rvert, \lvert\overline{R}^2\rvert\} = \max\{\lvert 30\rvert, \lvert 15\rvert\} =$ $\mathbf{30} \Rightarrow \overline{L}^2$. **Level 2:** $[\underline{D}^2, \overline{D}^2] = [-11, \mathbf{30}]$ $a \circ b = \{(-1-10)$ or $(-1+20)$ or $(10+20)$ or $(10\text{-}10)\}$ $= \mathbf{30} \Rightarrow a \circ b = \overline{L}^3 + \overline{R}^3$, $\Lambda^3 = \max\{\lvert\overline{L}^3\rvert, \lvert\overline{R}^3\rvert\} = \max\{\lvert 10\rvert, \lvert 20\rvert\} =$ $\mathbf{20} \Rightarrow \overline{R}^3$.	**Level 0:** $[\underline{D}^0, \overline{D}^0] = [-166, 451]$, $\Lambda^0 = \overline{D}^0$. $w(L^1) = 615$ and $w(R^1) = 2$. Hence, $\Lambda^1 = \max\{w(L^1), w(R^1)\} = \mathbf{615} = L^1$. **Level 1:** $[\underline{D}^1, \overline{D}^1] = [-165, 450]$ $w(L^2) = 41$ and $w(R^2) = 13$. $\Lambda^2 = \max\{w(L^2), w(R^2)\} = \mathbf{41} = L^2$. **Level 2:** $[\underline{D}^2, \overline{D}^2] = [-11, 30]$ $w(L^3) = 11$ and $w(R^3) = 10$. $\Lambda^3 = \max\{w(L^3), w(R^3)\} = \mathbf{11} = L^3$.

As a final remark on this example, we would like to mention that the two *2-best* parallel gradient based rules from the literature (Berner 1996) (Rules B/C) select x_2 and x_4 in parallel for re-partitioning this box. This results in a 10% lower reduction in the total pending status of all four siblings as compared to the reduction achieved by SIR-*bounds* and SIR-*widths*.

3.5 Convergence of SIR

First, we briefly summarize the major points in the convergence proofs. In the next two Lemmas, we show two exceptional subexpression forms where SIR may not be able to identify the source bounds at a given level k of the binary tree. The rules that deal with these exceptional cases are also described. It is shown that the latter rules ensure the convergence for SIR. Theorem 3.5 is the basic convergence proof for SIR.

The following Lemmas (Lemmas 3.1 and 3.2) discuss *even power, abs* and *trig* operators (*trig* denotes any trigonometric function) where SIR cannot label an interval bound at level $k+1$ symbolically if some ambiguous conditions hold on subexpression intervals at the relevant levels of the binary tree. Lemma 3.3 indicates two exceptional cases for interval multiplication operator.

Lemma 3.1 *Let the operator at any level k of a binary tree be* $\circ$ = " m" *(m is even) or* $\circ$ = *"abs", and let* $\Lambda^k = \underline{L}^k = 0$. *Further, let* $\underline{L}^{k+1} < 0$. *Then, SIR cannot identify* Λ^{k+1}.

Proof The proof is constructed by providing a counter example showing that SIR cannot identify Λ^{k+1} when the operator at level k is an even power and $\Lambda^k = 0$. Suppose that at level k we have the interval $[0, 16]$ and $\Lambda^k = \underline{L}^k = 0$. Let the operator at level k be 2. Since power is a unary operator, there is a single Left branch to this node at level $k+1$. Assume that the Left branch at level $k+1$ is a subexpression interval $[-4, 2]$. It is obvious that neither $\underline{L}^{k+1}$ nor $\overline{L}^{k+1}$ results in Λ^k. The case for the absolute value is similar. □

Lemma 3.2 *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* $\circ$ = *"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, SIR may not be able to identify* Λ^{k+1}.

Proof Similar to Lemma 3.1, a counter example is sufficient for a proof. Suppose we have the $\circ$ = "sin" operator at level k and the interval $[\underline{L}^k, \overline{L}^k] = [0.5, 1]$. Let the interval of the unary Left branch at level $k+1$ be $[\underline{L}^{k+1}, \overline{L}^{k+1}] = [\pi/6, 2\pi/3]$. Both $\underline{L}^{k+1}$ and $\overline{L}^{k+1}$ might result in $\underline{L}^k$ and none result in $\overline{L}^k$. The other stated cases can be proven similarly. □

Lemma 3.3 *Suppose the interval operator at a given level k is "*$\circ =' \times'$, *and,* $\underline{L}^{k+1}, \underline{R}^{k+1} < 0, \overline{L}^{k+1}, \overline{R}^{k+1} > 0, \left|\underline{L}^{k+1}\right| = \overline{R}^{k+1}$ *and* $\left|\underline{R}^{k+1}\right| = \overline{L}^{k+1}$. *Then, SIR might not be able to label a bound in the right or left sub-trees at level* $k+1$.

Proof It is sufficient to show a counter example for SIR's labeling procedure. Suppose '$\times$' type of interval operation exists at level k, with $\mathrm{L}^{k+1} = [-1, 2]$ and $R^{k+1} = [-2, 1]$. Then, at level k the $\times$ operator's interval is $[-4, 2]$. If the labeled bound is 2 at level k, then both $\underline{L}^{k+1} \times \underline{R}^{k+1} = \overline{L}^{k+1} \times \overline{R}^{k+1} = 2$ and we cannot choose among the two pairs of bounds at level $k+1$ that both provide the labeled bound at level k. □

Lemma 3.4 shows that SIR symbolically identifies the correct pair of candidate bounds resulting in Λ^k at any tree level k as long as the ambiguities indicated in Lemmas 3.1, 3.2 and 3.3 do not exist in a constraint expression.

Lemma 3.4 *For function expressions excluding the ambiguous subexpressions indicated in Lemmas 3.1, 3.2 and 3.3, SIR identifies the correct couple of bounds at level* $k+1$ *that result exactly in* Λ^k *at level k.*

Proof True by the monotonicity property of the remaining elementary interval operations and functions. □

3.6 Rules applied in case of labeling ambiguities

We now describe convergent rules that can be applied by *SIR_Tree* in case labeling ambiguities described in Lemma 3.1, Lemma 3.2 and Lemma 3.3 arise during tree traversal. Assume that there exists a subexpression of the type indicated in Lemma 3.1 at level k of a binary tree with $\Lambda^k = \underline{L}^k = 0$ and an interval bound at level $k+1, L^{k+1} < 0$. The bound labeling rule to be applied by *SIR_Tree* at level $k+1$ is $\Lambda^{k+1} = \underline{L}^{k+1}$. Now, assume that there exists a *trig* type subexpression at level k of a binary tree with *maxtrig* $\in [\underline{L}^k, \overline{L}^k]$ or *mintrig* $\in [\underline{L}^k, \overline{L}^k]$. The bound labeling rule to be applied by *SIR_Tree* at level $k+1$ is $\Lambda^{k+1} = \max\left\{|\underline{L}^{k+1}|, |\overline{L}^{k+1}|\right\}$. Finally, in the exceptional case found in Lemma 3.3, the choice in the two pairs of bounds is arbitrary.

Theorem 3.5 *The IPA algorithm is convergent both with the SIR-*bounds *and with the SIR-*widths *interval subdivision selection rules in the sense that the sequence of leading intervals converges only to global maximizer points.*

Proof Consider first the case when the SIR-*bounds* rule is applied. Assume that there exists such a subsequence $\{X_i\}$ of the leading boxes that $X_i \subset X_{i-1}$, and there exist a point x' in the search interval such that $f(x') < f(x^*)$, and x' is in each X_i. We demonstrate that it will imply a contradiction.

Prove first that during the subdivision in the subsequence $\{X_i\}$ every such variable that appears in the computation tree will be halved. It is so since otherwise when a variable that is used during computation would keep the original width while the widths of others converges to zero. As a consequence, then $\{X_i\}$ converge to a point regarding those variables that appear in the computed expression. This fact provides the contradiction, since the selection of the subinterval with the largest upper bound on the objective function cannot converge to a point x' in the search interval such that $f(x') < f(x^*)$, due to the α-convergence assumed.

For the case of the SIR-*widths* subdivision direction selection rule the proof is similar, but it is more straightforward. The respective interval subsequence has intervals the width of which converges to zero for all variables used within the computation.

However, the leading interval subsequences do not necessarily converge to points of the search space. It may happen that there is at least a variable that does not contribute to the objective function, i.e. it is not used in the computation tree. In cases where there is a continuum of global maximizer points and the resulting intervals highlight this phenomenon, such variables will keep their width in the original search interval. This is true for both introduced selection rules, and this indicates that these are as sophisticated as the rules B and C that also have this feature.

Note that the proposed interval subdivision direction selection rules can be well inserted into the directed acyclic graph framework developed by the COCONUT project (Schichl and Neumaier 2005).

4 Numerical experiments

4.1 Comparison basis

We compare the performance of SIR with two well established and efficient gradient-based subdivision direction selection rules (Rules B/C) from the literature (Ratschek and Rokne 1995; Csendes et al. 2000). These rules have become standard benchmarks because they have been identified as best performing among others after extensive testing. For a fair comparison with our multi-section approach, *Rules B/C* are also converted into multi-section rules by applying 2-*best* subdivision strategy (Berner 1996), i.e. the first two variables from the list (sorted according to Rules B/C) are partitioned. We describe these rules briefly below.

Rule B (Hansen 1992). Rule B chooses variables according to a maximal index consisting of variable interval width multiplied by the width of its respective first order derivative, $w(F_i'(X))$, i.e.

$$\text{Select} \quad x_k : C_k = \max_{i=1,\ldots,n} \{C_i\}, \quad \text{where } C_i = w(X_i)w(F_i'(X)). \tag{4.7}$$

Rule C (Ratz 1992). The first order derivative of each variable is multiplied by the difference between its interval and its midpoint, M_i. The variable with the maximum index value is selected by Rule C.

$$\text{Select} \quad x_k : C_k = \max_{i=1,\ldots,n} \{C_i\}, \quad \text{where } C_i = w(F_i'(X)(X_i - M_i)). \tag{4.8}$$

4.2 Test functions

27 well-known test functions from the literature are selected to compare performance of SIR against Rules B/C multi-section approach. The number of test instances becomes 34 as some functions such as Levy, Griewank and Schwefel are run with increasing number of dimensions (up to 30). The test functions are provided with their references and features in Table 1. The complexities and features of these test functions are discussed in detail in previous comparisons (e.g., Özdamar and Demirhan 2000) and they present a balanced portfolio from easy (such as Schwefel 3.1, Box), through moderate (e.g., Griewank) to difficult (e.g., Schwefel 3.7) problems with topological properties discussed in many global optimization references. (CUTEr—A Constrained and Unconstrained Testing Environment and IRIDIA: http://cuter.rl.ac.uk/cuter-www/problems.html.)

4.3 Results

Performance is measured in terms of the number of function and gradient calls (as indicated by FE and GE, respectively in Table 2), the CPU time in seconds, and the absolute deviation from the global optimum value. Positive absolute deviations occur in cases where methods fail to converge within 300 CPU seconds. The latter test instances are indicated at the end of Table 2. In SIR runs, FE does not include calls at subexpression levels because they are partial expression calls, and the latter are assumed as computational overhead. FE indicated for SIR is equal to the number of tree traversals. Rules B and C are supported by the monotonicity test since it does not require additional gradient calls. Finally, all methods use the cut-off test.

Table 1 Description and references of the test functions

Problem (Dimension)	Description	Reference	Name
Ackley (4)	Multimodal trigonometric function	Website MATLAB/TEST/Lazauskas	P5
Brownal (10)	Twice differentiable Sum of Squares.	CUTEr	P26
Box 3D (3)	Singular problem with manifold of solutions	Schwefel (1981)	P1
Cos 4 (4)	Multimodal trigonometric	Breiman and Cutler (1993)	P6
Dixon3dq (10)	Quadratic function	CUTEr	P24
Djong's Function 2 (8)	Long flat valley	De Jong (1975)	P20
Eg1 (3)	Twice differentiable trigonometric function	CUTEr	P2
Exp 6 (6)	Exponential function	Brieman and Cutler (1993)	P14
Extended Kearfott (4)	Polynomial function	Kearfott (1979)	P7
Extrosnb (10)	Twice differentiable Sum of Squares.	CUTEr	P23
Genhumps (5)	Twice differentiable Sum of Squares.	CUTEr	P13
Griewank (5, 10, 20)	Wide spread regularly distributed maxima, trig.	IRIDIA	P10, P22, P28
Hartman (6)	4 local minima	Törn and Zilinskas (1989)	P32
Hs045 (5)	Twice differentiable geometric function	CUTEr	P9
Levy 14,16,18 (3, 5,7)	2700, 10^5, 10^8 local minima, trigonometric	Levy et al. (1982)	P3, P12, P17
Levy 10,11,12 (5, 8, 10)	10^5, 10^8, 10^{10} local minima, trigonometric	Levy et al. (1982)	P11, P19, P25
Michalewicz (5)	Multimodal trigonometric	IRIDIA	P31
Powell (4)	Singular, Hessian at origin	Moré et al. (1981)	P4
Rastrigin (8)	Highly multimodal trigonometric, regularly distributed local maxima	Website MATLAB/TEST/Lazauskas	P18
Rosenbrock (10)	Long curved only slightly decreasing valley	Rosenbrok (1970)	P21
S271 (6)	Quadratic function	Schittkowski (1987)	P15
S288 (20)	Quadratic function	Schittkowski (1987)	P33
Schwefel 1.2 (4)	Continuous unimodal	Schwefel (1981)	P8
Schwefel 3.1 (3)	Unimodal function	Schwefel (1981)	P34
Schwefel 3.7 (15, 30)	Singular Hessian at $x^* = 0$	Schwefel (1981)	P27, P29
Shekel (4; $m = 10$)	Multimodal test function	Törn and Zilinskas (1989)	P30
Sphere (7)	Unimodal	IRIDIA	P16

Table 2 Comparison of numerical results

Problem	SIR-*Bound*		SIR-*widths*		Rule B			Rule C		
	FE	CPU	FE	CPU	FE	GE	CPU	FE	GE	CPU
P1	180	0.407	164	**0.296**	**140**	141	0.313	**140**	141	0.312
P2	414	0.203	292	0.125	**12**	13	**0.016**	**12**	13	**0.016**
P3	324	0.142	**156**	**0.095**	164	165	0.172	164	165	0.188
P4	1224	0.875	1224	**0.859**	**1000**	1004	1.141	1060	1061	1.297
P5*	4260	2.950	**416**	**0.312**	–	–	–	–	–	–
P6	5460	6.771	4664	4.263	**276**	277	**0.391**	**276**	277	**0.391**
P7	**296**	**0.125**	**296**	0.156	432	433	0.421	432	433	0.421
P8	1488	5.500	260	**0.125**	**200**	201	0.140	**200**	201	0.140
P9	320	0.375	512	0.530	**20**	21	**0.015**	20	21	0.047
P10	316	0.359	252	**0.234**	**240**	241	0.390	**240**	241	0.391
P11	316	0.453	304	**0.423**	**236**	237	0.719	**236**	237	0.874
P12	416	0.469	340	**0.155**	**268**	269	0.594	**268**	269	0.594
P13	10556	11.384	496	**0.593**	**416**	417	1.155	**416**	417	1.155
P14	624	0.671	572	0.514	**28**	29	**0.062**	**28**	29	**0.062**
P15	932	0.719	524	**0.344**	**520**	521	0.781	**520**	521	0.781
P16	384	0.281	384	0.281	**108**	109	**0.203**	**108**	109	**0.203**
P17	416	**0.500**	532	0.765	**364**	365	1.624	**364**	365	1.624
P18	538	1.187	492	**0.765**	**488**	489	2.140	**488**	489	2.140
P19	568	**1.311**	580	1.483	**380**	381	3.156	**380**	381	3.156
P20	488	**0.717**	488	0.750	**484**	485	2.219	488	489	2.220
P21	**652**	1.410	**652**	**1.389**	720	721	6.156	708	709	6.047
P22	640	2.017	572	**1.110**	488	489	3.578	**484**	485	3.484
P23	572	**1.141**	572	1.145	552	553	4.437	**544**	545	4.338
P24	616	**1.297**	616	1.329	644	655	3.859	**588**	589	3.687
P25	672	1.915	564	**1.529**	**472**	473	6.422	**472**	473	6.422
P26	648	3.393	608	**2.184**	**484**	485	5.422	**484**	485	5.516
P27	128	**0.172**	128	0.203	**124**	125	1.313	**124**	125	1.313
P28	1332	13.590	1120	**5.891**	**960**	961	39.482	972	973	39.733
P29	252	**0.812**	252	0.830	**244**	245	17.187	**244**	245	17.124
Average	1208	2.108	622	0.989	374	375	3.697	455	456	3.703
SD	2141	3.283	815	1.266	255	256	7.832	259	259	7.865
Best	–	8	–	16	–	–	5	–	–	4

Problems not converged within 300 CPU seconds

Problem	SIR-bound		SIR-widths		Rule B			Rule C		
	FE	Deviation	FE	Deviation	FE	GE	Deviation	FE	GE	Deviation
P30	10712	0.008	10656	**0.000**	17182	17183	**0**	17182	17183	**0**
P31	10112	2.884	14292	**0.000**	17506	17507	**0**	17274	17275	**0**
P32	9636	0.163	12144	0.002	9484	9485	**0**	9484	9485	**0**
P33	1356	**0.000**	135	**0.000**	6196	6197	3000	10576	10577	3000
P34	308	**0.000**	21072	**0.000**	220	221	**0**	220	221	**0**

* problem with computing the gradient value

A run is completed when for all non-discarded pending boxes the difference of the function upper bound over the box to the current lower bound is less than 1×10^{-13}. The runs were executed on a PC with 2 GB RAM, 2.4 GHz Intel Xenon CPU, under Windows OS system. All codes were developed with Visual C++ 6.0 interfaced with the PROFIL interval arithmetic library.

In the last 5 rows of Table 2, we can observe that Rule B and C were not able to converge on 4 test functions within the CPU time limit imposed, but they are able

Table 3 Total CPU times in seconds for small and larger size problems

Dimension	SIR-*bounds*	SIR-*width*	Rule B	Rule C
$n < 5$	16.973	6.231	2.594	2.765
$n \geq 5$	44.173	22.447	100.914	100.911

to converge for the 5th one in 0.141 seconds. Similarly, the SIR-*bounds* rule does not converge for the first 3 functions, but it was able to converge for the 4th and 5th functions within 6.153 seconds, and 0.282 seconds respectively. SIR-*widths* does not converge for first 3 test functions and the 5th test function, but it was able to converge for 4th one within 6.374 seconds. The performance of SIR is notable for the function S288 where Rules B/C end up very far from the global optimum.

Considering all 34 test functions, the results obtained by Rules B and C are not significantly different. When the first part of Table 2 is analyzed, we observe that the average number of function calls for SIR is larger than those of Rules B and C (including their gradient calls). Despite this fact the average CPU time required for SIR-*bounds* is almost half of those of Rules B and C. That of SIR-*widths* is almost one-fourth of Rules B/C. The tree traversal overhead in SIR is comparable with the task of calculating the gradient in the other rules. The number of best solutions obtained by SIR-widths compares very well with others. Hence, we can conclude that SIR's symbolic methodology of selecting the maximum impact variables is more efficient than that of the function rate of change based rules.

In Table 3, we provide a summary of total CPU times taken by all rules for functions with less than five dimensions and for those greater than 5 dimensions. In the first part of Table 3, we observe that SIR's performance is inferior in test functions up to 5 dimensions. In problems with larger dimensions, its performance is significantly superior as compared to Rules B and C. When the outlier CPU time (Griewank 20D) was removed from this set, we have found the difference in performance statistically significant (at a 5% significance level). In Table 3, the total CPU time needed by all three methods is given for the first 29 test problems (split into less than or greater than 5 dimensions) where all methods converge. This outcome is expected because the sequence of variables to be partitioned gains more importance in larger dimensional problems. Both Rules B and C are affected by the width of variable domains, and this tends to push the selected variable sequence into a more balanced manner in terms of box size. However, the size of variable domains has a more implicit impact on the choice of variables in SIR.

5 Conclusion

A new SIIA has been developed to improve the convergence rate in IPA. The proposed subdivision direction selection rule, SIR stems from SIIA. SIIA is based on parsing a function into its sub expressions, converting it into a binary tree where every subexpression is a node, and calculating their interval contributions to the total function range. SIR is a labeling procedure that traverses the sub expressions tree to identify a pair of maximum impact variables. The impact of the variables need not be quantified in this approach. Hence, the inherent uncertainty that exists in interval gradient ranges is eliminated in SIIA. SIR targets a reduction in the overestimation of a parent box's function range with its variable selection scheme.

Two versions of SIR are proposed here: SIR-*bounds* and SIR-*widths*. While the first version identifies and labels the maximum impact interval bounds at subexpression levels, the second version labels subexpressions with largest interval widths. The labeling procedure *SIR_Tree*, traverses through labeled subexpressions and finally reaches the maximum impact variables in the function expression.

SIR's efficiency is illustrated by numerical tests and compared with function rate of change based rules from the literature. It is also possible to utilize SIR in any IPA that is used in the fields of constrained optimization (COP) and continuous constraint satisfaction problems (CCSP). Currently, work is conducted to improve the solvability of standard CCSP using SIIA.

Acknowledgements The present work has been partially supported by the grants OTKA T 048377, and T 046822. The authors are grateful to Hermann Schichl for his valuable comments and suggestions.

References

Berner, S.: New results on verified global optimization. Computing **57**, 323–343 (1996)
Breiman, L., Cutler, A.: A deterministic algorithm for global optimization. Math. Program. **58**, 179–199 (1993)
Casado, L.G., Martinez, J.A., García, I.: Experiments with a new selection criterion in a fast interval optimization algorithm. J. Global Optimiz. **19**, 247–264 (2001)
Ceberio, M., Granvilliers, L.: Solving nonlinear systems by constraint inversion and interval arithmetic. Lecture Note Arti. Intell. **1930**, 127–141 (2000)
Csallner, A.E., Csendes, T., Markót, M.Cs.: Multi-section in interval branch and bound methods for global optimization I. Numerical Tests. J. Global Optimiz. **16**, 219–228 (2000a)
Csallner, A.E., Csendes, T., Markót, M.Cs.: Multi-section in interval branch and bound methods for global optimization II. Theoretical Results. J. Global Optimiz. **16**, 371–392 (2000b)
Csendes, T., Klatte, R., Ratz, D.: A Posteriori Direction Selection rules for interval optimization methods. Central Eur. J. Operation Res. **8**, 225–236 (2000)
Csendes, T., Ratz, D.: A review of subdivision selection in interval methods for global optimization. ZAMM Z. Angew. Math. Mech. **76**, 319–322 (1996)
Csendes, T., Ratz, D.: Subdivision direction selection in interval methods for global optimization. SIAM J. Numer. Anal. **34**, 922–938 (1997)
De Jong K.: An analysis of the behaviour of a class of genetic adaptive systems. PhD Thesis, University of Michigan, Ann Arbor (1975)
Granvilliers, L.: An interval component for continuous constraints. J. Comput. Appl. Math. **162**, 79–92 (2004)
Granvilliers, L., Monfroy, E., Benhamou, F.: Symbolic-interval cooperation in constraint programming. Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation. London, Canada (2001)
Hammer, R., Hocks, M., Kulish, U., Ratz, D.: Numerical Toolbox for Verified computing I Springer-Verlag, Berlin (1993)
Hansen, E.: Global optimization using interval analysis. Marcel Dekker, New York (1992)
Jansson, C., Knüppel, O.: A branch and bound algorithm for bound constrained global optimization. J. Global Optimiz. **7**, 297–331 (1995)
Kearfott, B.: An efficient degree-computation method for a generalized method of bisection. Numeric. Math. **32**, 109–127 (1979)
Kearfott, B., Kreinovich, V.: Applications of Interval Computations, Applied Optimization. Kluwer, Dordrecht, The Netherlands (1996)
Knüppel, O.: PROFIL/BIAS—A fast interval library. Computing **53** ,277–287 (1994)
Levy, A.V., Montalvo, A., Gomez, S., Calderon, A.: Topics in global optimization. Lecture Note. Math. **909**, 18–33 (1982)
Lhomme, O., Gotlieb, A., Rueher, M.: Dynamic optimization of interval narrowing algorithms. J. Logic Program. **37**, 165–183 (1998)
Moore, R.E.: Interval Analysis. Prentice-Hall, Englewood Cliffs, NJ (1966)
Moré, J.J., Garbow, B.S., Hillstrom, K.E.: Testing unconstrained optimization software. ACM Trans. Math. Software **7**, 17–41, (1981)

Neumaier, A.: Interval Methods for Systems of Equations, Encyclopedia of Mathematics and its Applications, 37. Cambridge University Press, Cambridge (1990)
Özdamar, L., Demirhan, M.: Experiments with new probabilistic search methods in global optimization. Comput. Operations Res. **27**, 841–865 (2000)
Pintér, J.: Convergence qualification of adaptive partitioning algorithms in global optimization. Math. Program. **56**, 343–360 (1992)
Ratschek, H., Rokne, J.: Interval methods. In: Horst, R., Pardalos, P.M. (eds.), Handbook of Global Optimization. Kluwer, Dordrecht, 751–828 (1995)
Ratschek, H., Rokne, J.: New Computer Methods for Global Optimization. John Wiley, New York (1988).
Ratz, D.: Automatische Ergebnisverifikation bei globalen Optimierungsproblemen. Dissertation, Universität Karlsruhe, Germany (1992)
Ratz, D., Csendes, T.: On the selection of subdivision directions in Interval Branch-and-Bound Methods for Global Optimization. J. Global Optimiz. **7**, 183–207 (1995)
Rosenbrock, H.H.: State-Space and Multivariable Theory. Wiley, New York (1970)
Sam-Haroud, D., Faltings, B.: Consistency techniques for continuous constraints. Constraints **1**, 85–118 (1996)
Schichl, H., Neumaier, A.: Interval analysis on directed acyclic graphs for global optimization. J. Global Optimiz. **33**, 541–562 (2005)
Schittkowski, K.: More Test Examples for Nonlinear Programming Codes. Lecture Notes in Economics and Mathematical Systems, vol. 282. Springer-Verlag, Berlin (1987)
Schwefel, H.P., Numerical Optimization of Computer Models. Wiley, New York, (1981)
Törn, A., Žilinskas, A., Global Optimization. Lecture Notes in Computer Science, vol. 350. Springer-Verlag, Berlin (1989)