Heuristic Rejection in Interval Global Optimization[1]

L.G. Casado[2], I. García[3], T. Csendes[4], and V.G. Ruíz[5]

Communicated by L.C.W. Dixon

---

[1]This work was supported by the Ministry of Education of Spain (CICYT TIC99-0361) and by Grants FKFP 0449/99, OMFB ES 24/2001, OTKA T 017241, T 32118, and T 034350.

[2]Professor, Department of Computer Architecture and Electronics, University of Almería, Almería, Spain

[3]Professor, Department of Computer Architecture and Electronics, University of Almería, Almería, Spain

[4]Associate professor, Institute of Informatics, University of Szeged, Szeged, Hungary

[5]Teaching assistant, Department of Computer Architecture and Electronics, University of Almería, Almería, Spain

Abstract. Based on the investigation carried out (Ref. 1), this paper incorporates new studies about the properties of the inclusion functions on subintervals while a branch-and-bound algorithm is solving global optimization problems. It is found that the relative place of the global minimum value within the inclusion function value of the objective function at the current interval indicates mostly whether the given interval is close to a minimizer point. This information is used in a heuristic interval rejection rule that can save a big amount of computation. Illustrative examples are discussed and an extended numerical study shows the advantages of the new approach.

Key Words. Interval method, global optimization, branch-and-bound algorithms.

1 Introduction

In this paper we investigate a heuristic variant of a simple interval Branch and Bound algorithm which allows fast solutions for some of the above mentioned very hard global optimization problems to be obtained. Madsen and Zertchaninov reported a stochastic, heuristic version of the interval optimization algorithm that aimed also to solve high memory complexity problems (Ref. 2). The suggested technique was a kind of garbage collection procedure handling the remaining sub-boxes. In Ref. 3, a local search method based on interval analysis information is described. The proposed method was tested for a wide set of hard functions and the global solution was reached when enough memory resources were devoted. A heuristic solution has also been proposed by Casado et al. in Ref. 1, where the algorithm follows the framework of the Moore-Skelboe algorithm (Ref. 4) with the midpoint test (Ref. 5) and, additionally, includes a heuristic rejection criterion (HRC). This heuristic rejection criterion is based on the value of a parameter named *reject index* which is evaluated for every tested interval.

The method investigated in the present work is an improved version of the algorithm of Casado et al. (Ref. 1) where the monotonicity test (Refs. 6–8) has been added and the heuristic rejection criterion has been slightly modified by including an additional constraint. The resulting heuristic algorithm cannot ensure that the solution includes all the global optimizer points, but our experimental results show that it works successfully for all the tested functions, including several hard problems.

2 General Framework and Notations

Consider the bound constrained global optimization problem (Ref. 9):

$$\min_{x \in X} f(x), \qquad (1)$$

where the n-dimensional interval $X \subseteq \mathbb{R}^n$ is the search region, and $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function. The global minimum value of f is denoted by f^*, a global minimizer point by x^*, and the set of global minimizer points of f on X by X^*, i.e.:

$$f^* = \min_{x \in X} f(x) \qquad \text{and} \qquad X^* = \{x \in X \mid f(x) = f^*\}.$$

Real numbers are denoted by $x, y, \ldots$, and real bounded and closed interval vectors by $X = [\underline{X}, \overline{X}], Y = [\underline{Y}, \overline{Y}], \ldots$, where $\underline{X}_i = \min\{x \in X_i\}$ and $\overline{X}_i = \max\{x \in X_i\}$, are the lower and upper bounds, respectively, for $i = 1, 2, \ldots, n$. The set of compact intervals is denoted by $\mathbb{I} := \{[a, b] \mid a \leq b,\ a, b \in \mathbb{R}\}$ and the set of n-dimensional interval vectors (also called boxes) by $\mathbb{I}^n$. $\mathbb{I}^n(X)$ will denote those Y intervals in $\mathbb{I}^n$

for which $Y \subseteq X$. For real and interval vectors the notation

$$x = (x_1, x_2, \ldots, x_n)^T,\ x_i \in I\!R \text{ and } X = (X_1, X_2, \ldots, X_n)^T,\ X_i \in I\!I$$

is used ($i = 1, 2, \ldots, n$). The width of the interval X is defined by $w(X) = \overline{X} - \underline{X}$ (if $X \in I\!I$), and $w(X) = \max_{i=1}^{n} w(X_i)$, if $X \in I\!I^n$. The midpoint is defined by $m(X) = (\underline{X} + \overline{X})/2$, if $X \in I\!I$, and $m(X) = (m(X_1), m(X_2), \ldots, m(X_n))^T$, if $X \in I\!I^n$.

Our implementation is based on the Moore-Skelboe interval B&B algorithm (Ref. 4). For interval B&B algorithms the Bounding is given by interval arithmetic which allows an inclusion function to be obtained. A function $F : I\!I^n \to I\!I$ is called an *inclusion function* of f in $X \subseteq I\!R^n$, if $x \in X$ implies $f(X) \subseteq F(X)$, where $f(X)$ is the range of f on X.

The most common Branching rule consists of bisecting boxes orthogonal to the coordinate with the largest width. Based on the proposals of Ratz and Csendes (Ref. 10), Berner (Ref. 11) has reported better results using bisection in two dimensions in one step. Hansen (Ref. 7) proposed subdivision into three sub-intervals. In the present work bisection in two dimensions in one step is applied because some of the test functions used in this work are of large dimension.

The best-first selection rule has proven to be more efficient than the oldest-first and the depth-first strategies (Ref. 11). In this work, following the best-first strategy, boxes are sorted in a non-decreasing order with respect to $\underline{F}(Y)$ as the first ordering criterion and in decreasing order with respect to the age of the boxes as the second ordering criterion. In the present case $w(Y) < \epsilon$ was required as a stopping criterion for all the remaining boxes.

The simplest elimination rule is the cut-off test, where boxes Y such that $f^* \leq \tilde{f} < \underline{F}(Y)$ are rejected ($\tilde{f}$ is the current verified upper bound of the minimum). In addition to the cut-off test, in our implementation the middle point test as proposed by Ichida and Fujii (Ref. 5), the monotonicity test (Refs. 6–8), and a new heuristic rejection criterion described in the following sections are used.

The basic algorithm has the input parameters denoted as:

X : interval search region,

f : real objective function,

F : interval inclusion function of f,

$\tilde{f}$: updated best upper bound of f^*,

$\tilde{x}$: $\overline{F}(\tilde{x}) = \tilde{f}$.

ϵ : stopping criterion parameter $(w(Y) < \epsilon)$,

T : work tree,

Q : final tree, and

$+/-$: denote include/discard boxes in a tree.

The basic algorithm starts by initializing the work tree, T, with a box containing the search region. The final tree, Q, is initialized as an empty set and the current upper bound of the minimum $(\tilde{f})$ is set to the value of the upper bound of the inclusion function for the initial search region. The Selection rule is applied to the work tree choosing the first box B of the tree T, for which $\underline{F}(B) = \min_{B^i \in T} \underline{F}(B^i)$. The middle point test is carried out on the selected box and if $\tilde{f}$ is improved, the cut-off test is applied to the work and final trees. Then, the branching rule is executed on the selected box and the generated sub-boxes ($B^1, B^2, \ldots, B^s$, in this work $s = 4$) can be rejected by the traditional elimination rule, or the monotonicity test. Otherwise, they are stored in the final tree, if they fulfill the termination criterion, or in the work tree. The procedure is repeated until the work tree is empty.

3 Properties of the New *Reject Index* Parameter

Suppose that for a given function $f : X \subseteq I\!R^n \rightarrow I\!R$, f^*, or an approximation of f^* ($\tilde{f}$) is a priori known. It may be obtained for example using a local search algorithm or during the execution of the B&B algorithm itself. Let us define a parameter $pf^*(Y)$ called *reject index* as: $pf^*(Y) = (f^* - \underline{F}(Y))/(\overline{F}(Y) - \underline{F}(Y)) \in [0, 1],\ Y \subseteq X$.

This parameter was designed mainly based on the following: Traditionally an interval Y with the minimal value of $\underline{F}(Y)$ was considered as the best candidate to contain a global minimum. However, usually the larger the interval Y, the larger the over-estimation of the range $f(Y)$ obtained in $F(Y)$. Therefore a box could be considered as a good candidate to contain a global minimum just because it is larger than others. In order to compare sub-intervals with different size fairly we normalize the distance between f^* and $\underline{F}(Y)$.

The idea behind pf^* is simply that we expect the over-estimation to be symmetric, i.e. the over-estimation $\overline{F}(Y)-\overline{f}(Y)$ above $f(Y)$ is closely equal to the over-estimation $\underline{f}(Y) - \underline{F}(Y)$ below $f(Y)$, for small sub-intervals containing a global minimizer point (that is at the same time a stationary point). Hence, for such intervals Y the relative place of the global optimum value inside the $F(Y)$ interval should be high, while for intervals far from global minimizer points pf^* must be small. Obviously, there are exceptions, and there exist no theoretical proof that pf^* would be a reliable indicator of nearby global minimizer points.

In this section the relation between the value of $pf^*(Y)$ and the closeness of Y to an $x^* \in X^*$ is tested. Our experiments show that for large $pf^*(Y)$ value the interval Y is usually close to a local or global minimizer point.

The *reject index* has been previously used in parallel interval $B\&B$ algorithms as a predictor of the computational work associated to boxes in the work tree. In Ref. 12, it has been shown that using *reject index* an almost perfect work load balance for parallel implementations of the Interval $B\&B$ algorithms can be obtained. According to our experiments a big *reject index* value for the interval Y indicate that it is usually close to a local or global minimizer point (Ref. 13).

Definition 3.1 F is said to be an inclusion isotone function over X if $\forall Y, Z \in I\!I^n(X)$ $Y \subseteq Z$ implies $F(Y) \subseteq F(Z)$.

Definition 3.2 *F is an order α inclusion function* (α-convergent) *of f over X if $\forall Y \in I\!I^n(X)$ $w(F(Y)) - w(f(Y)) \leq Cw(Y)^\alpha$, where C and α are positive constants.*

Theorem 3.1 Assume that for an optimization problem (1) the inclusion function $F(X)$ of $f(x)$ is isotone, and α-convergent with given positive constants C and α. Assume further that the *reject index* parameter pf^* is less than 1 for all the sub-intervals of X. Then an arbitrary large number $N(>0)$ of consecutive leading intervals of the list T in the basic algorithm may have the properties that:

(i) neither of these processed intervals contains a stationary point, and

(ii) during this phase of the search the pf^* values are maximal for these intervals as compared with the subintervals of the current working list.

Proof. Consider an optimization problem (1). Assume that its inclusion isotone function $F(X)$ is α-convergent with certain C and α positive constants, there exists a sub-interval $Y \subseteq X$ from the working tree such that $\underline{F}(Y) > f^* = \min_{x \in X} f(x)$, and Y does not contain any stationary points. There exist such sub-intervals in the search region of most sensible global optimization problems.

Construct now a new inclusion function $G(Z) = [\underline{F}(Z) - Dw(Z)^{\alpha}, \overline{F}(Z)]$ for intervals $Z \subseteq Y$ and $Z \supseteq Y$, and $G(Z) = F(Z)$ otherwise, where D is a positive constant. It is easy to see that $Z \subseteq Z'$ implies $G(Z) \subseteq G(Z')$, i.e. G is isotone. On the other hand, G is also α-convergent with the constants $C + D$ and α.

Consider the iteration step when the interval Y was generated and decided not to be selected for further subdivision. Denote the originally selected sub-interval by X'. Then $\underline{F}(X') \leq \underline{F}(Y)$. We set the value of D in the definition of $G(X)$ in such a way, that if the basic algorithm utilizes the inclusion function $G(X)$, then the current intervals will be selected in N consecutive steps within Y, and also the pf^* values will be higher for these intervals than for X'. Let us see how to set D.

Notice, that as we increase the value of D, the respective pf^* value converges to 1 from below. Denote the *reject index* parameter calculated for the inclusion function $G(X)$ by pf^*_G, and that for $F(X)$ by pf^*_F. Now it is obvious that with a properly chosen constant D, both the $\underline{G}(Z) < \underline{F}(X')$ and also for the *reject index* parameter value the $pf^*_G(Z) > pf^*_F(X')$ relations hold for any $Z \subseteq Y$ interval ($pf^*(X') < 1$ as assumed).

We can select an arbitrary series of N nested intervals within Y, and choose the maximal constant D. With this value the original optimization problem with the new inclusion function of $G(X)$ ensures the properties 1 and 2 of Theorem 3.1. $\square$

The consequence of Theorem 3.1 is that even with an isotone and α-convergent inclusion function, the generated sequence of investigated sub-intervals can be distracted for an arbitrarily long time from the global minimizer points. The assumption that the *reject index* parameter pf^* is less than 1 for all the sub-intervals of X holds for all relevant practical problems. It could be hurt only in such problems, where the objective function value is constant in a full sub-interval Y of the search region, its value is equal to the global minimum, and also $\overline{F}(Y) = f^*$. In such extremely rare cases, we would obtain a more than satisfactory result (like Y and $F(Y)$) quickly. The outside rounding usual in computer implementation makes this case also impossible.

4 Improving a Branch-and-Bound Algorithm

We investigate the behavior of the pf^* value of a box compared to that of its offsprings (sons and grandsons). Figure 1 shows several examples of possible behavior forms of pf^* for a box and its offsprings which happened in our experiments. In the upper left picture, values of pf^* are shown for a box and its offsprings obtained in the first stage of the algorithm execution and an irregular behavior can be seen in the pf^* value of the offsprings.

The first box at the upper right picture is the same box that the last one from the upper left picture, containing a minimizer point. Now a more regular behavior in the pf^* values of the offsprings occurs. Only those boxes with the larger pf^* values contain global minimizer points. In the lower pictures, boxes generated at the last iterations of the algorithm are shown. In the left picture, the first box contains a global minimizer point and only its sons and grandsons with higher pf^* values have minimizer points. Notice also that the differences between the pf^* values are very small (< 0.05). On the right hand side, the initial box does not contain any global minimizer point, and in this case the value of pf^* decreases significantly for all its sons and grandsons. Extensive experiments on a wide set of test functions have shown the same results.

Increasing *reject index* values mean that the relative position of the f^* value within the interval of $F(Y)$ is higher and higher. It can indicate that the average function value of a point within those intervals is decreasing, which is in accordance with the assumption that this sequence of intervals approaches to a minimizer point. Also, the preliminary numerical studies confirmed that increasing *reject index* values were rarely connected to interval sequences converging to points with function values above f^*.

Our proposal of a new B&B global optimization algorithm consists of the basic algorithm with a new rejection criterion. During the execution of the B&B algorithm, we distinguish two different stages. The initial stage, if it exists, finishes when an increase of the pf^* value appears. During this stage the new algorithm works as the basic one. For the second stage we have stated that when the pf^* value of a box (B''), its son (B') and its grandson (B) consecutively decreased, then the box B does not contain a global optimizer point, and therefore it can be rejected. B and B' are obtained from the subdivision of B' and B'', respectively; thus $B \subset B' \subset B''$. In the new algorithm the difference between the pf^* values of B'' and B must be greater than a threshold ($DecTh$). Notice that the larger the $DecTh$ value the smaller the number of boxes eliminated by the new rejection criterion and $DecTh = 1.0$ inhibits the rejection. Using very small $DecTh$ values we could reject boxes containing minimizer points and with $DecTh = 0.0$ only sequences of increasing pf^* values are kept. This new rejection criterion does not allow to eliminate a box which contains the point $\tilde{x}$ from which the current value of $\tilde{f}$ was obtained.

Theorem 4.1 For each global optimization problem (1) there exist an infinity of inclusion functions that ensure the convergence of the new algorithm to the set of global minimizer points. For interval sequences converging to stationary points of a smooth objective function these inclusion functions are α convergent with $\alpha = 2$.

Proof. The proof will construct a class of inclusion functions step by step. Consider first the inclusion function class for given constants $a_i, b_i \geq 0$, $(i = 0, 1, \ldots)$ and with $\underline{f}(Y)$ as the minimum of the range of $f(y)$ on Y

$$F(Y) = [\underline{f}(Y) - a_i\, w(f(Y)), \overline{f}(Y) + b_i\, w(f(Y))].$$

The index i gives the number of subdivisions made to obtain the interval Y: e.g. $i = 0$ for $Y = X$, and $i = 1$ for those sub-intervals which are the results of the first subdivision. $F(Y)$ is an inclusion function of $f(y)$, since $y \in Y$ implies $f(y) \in F(Y)$.

The ideal case, $a_i = b_i = 0$ is not usual for interval arithmetic based inclusion functions, however $a_i = 0$ may occur even with finite length data representation: e.g. when the objective function has the form of $f(y) = (g(y))^2 + c$ where $c \in \mathbb{R}$.

Consider now a sub-interval $Y \subset X$ that contains a global optimizer point y^*. The *reject index* value for this interval is

$$pf^*(Y) = \left(f^* - (\underline{f}(Y) - a_i w(f(Y)))\right) / \left(\overline{f}(Y) + b_i w(f(Y)) - (\underline{f}(Y) - a_i w(f(Y)))\right).$$

With $\underline{f}(Y) = f^*$ and $\overline{f}(Y) - \underline{f}(Y) = w(f(Y))$ we obtain $pf^*(Y) = (a_i)/(1 + a_i + b_i)$. Now non-decreasing pf^* values inhibit the deletion by the new rejection rule, and these can be achieved easily, e.g. by setting $a_i = a$, $b_i = b$ constant, or by allowing b_i to increase slightly (while $a_i = a$ is constant). With such a setting of a_i and b_i, the new rejection criterion cannot delete subintervals containing global minimizer points. Inclusion functions satisfying these conditions form the aimed class. Notice that the construct has not utilized the necessary condition of $\tilde{x} \notin B$ of a deletion.

For the case $a_i = 0$, all the current intervals will be exclusively those containing a global minimizer point, since only for these intervals $\underline{F}(Y)$ can attain the minimal lower bound value of f^*.

For a smooth function $f(y)$ the convergence order can be calculated by $w(F(Y)) - w(f(Y)) = (\overline{f}(Y) + b_i\, w(f(Y))) - (\underline{f}(Y) - a_i\, w(f(Y))) - w(f(Y)) = (a_i + b_i)w(f(Y))$. For bounded a_i, b_i, it is less than or equal to $Cw(f(Y)) \leq C(f'(c)w(Y) + |H(X)|w(Y)^2)$, where $c \in Y$, and $|H(X)|$ is the maximal norm of the Hessian of f on X. When Y contains a stationary point x', we can set $c = x'$, and then $f'(c) = 0$. In other words, the inclusion function will be α-convergent with $\alpha = 2$ for interval sequences converging to stationary points. Otherwise the order of convergence is linear. $\square$

Although the class of inclusion functions constructed in the proof of Theorem 4.1 may incorporate inclusion functions generated by natural interval extension, the earlier ones have no practical significance since the exact range of the underlying objective function is needed to build them. Still, the proof shows which properties of the inclusion functions can ensure convergence for the new algorithm. The defined inclusion functions overestimate the range in a similar way and with proper parameter setting to a similar extent as natural interval extensions do. Although the quadratic convergence was proven only for interval sequences converging to stationary points, it is the typical case when the inclusion functions are used within an optimization algorithm.

Theorem 4.2 Consider a global optimization problem (1) that has a unique global minimizer point x^*. Then there exists an isotone and α-convergent inclusion function $F(X)$ of $f(x)$ such that the new algorithm does not converge to x^*.

Proof. Consider a problem satisfying the conditions of the theorem, and the first four intervals of the interval sequence $\{X_i\}$ that converge to $[x^*, x^*]$: $X_0 = X$, X_1, X_2, and X_3. Assume that the original inclusion function $G(X)$ is isotone and α-convergent. Such an inclusion function obviously exists, it can be generated e.g. by the natural interval extension (Ref. 4). The $pf^*(X_i)$ value can be set to any given value between zero and one by decreasing the lower and increasing the upper bounds of the inclusion function value. If we alter only these four inclusion function values, i.e. $F(Y) = G(Y)$ for all other sub-intervals, then the α-convergence property cannot be hurt. In a similar way, we can keep the isotonicity, if we set the new $F(X_i)$ values in such a way, that $F(X_i) \supseteq F(X_{i+1})$ holds for $i = 0, 1, 2,$ and 3. Notice that the same $pf^*(X)$ value can be achieved by increasing the difference of $\underline{F}(Y)$ and $\overline{F}(X)$ properly. For $DecTh$ $= 0.04$ now set the $F(X_0), \ldots, F(X_3)$ values in such a way that $pf^*(X_0) = 0.98$, $pf^*(X_1) = 0.99$, $pf^*(X_2) = 0.96$, and $pf^*(X_3) = 0.94$.

By the construction, the initial stage finishes when X_1 is processed, and the interval X_3 will be fathomed since the pf^* values are decreasing for the last three sub-intervals, and also the difference in the pf^* values is larger than the set 0.04 threshold value regarding X_1 and X_3. We can also assume that the objective function value at the midpoint of X_3 does not allow an update of $\tilde{f}$ (it can be achieved e.g. by setting $\overline{F}(m(B')) := \overline{F}(m(B)))$. Hence the interval X_3 will not be processed further, and in this way the new algorithm cannot converge to the point $x^* \in X_3$. Since for all the four selected intervals $\underline{F}(X_i) \leq \underline{G}(X_i)$ holds, the respective intervals will be selected not later than the same algorithm would do for $G(Y)$. $\square$

Although Theorem 4.2 investigates only the case when there exists a unique global minimizer point in the search region of a global optimization problem, a similar construct can be used for problems with larger (finite) number of global minimizer points. The proof of Theorem 4.2 also indicates that although there may exist real life problems for which the new algorithm does not converge to global minimizer points, this behavior still requires a special constellation of the related inclusion function values.

Figure 2 shows the pf^* values sequentially generated during the execution of the basic and the new algorithm for the Goldstein-Price test function. Boxes with $pf^* < 0$ have not been drawn. In both graphs of Figure 2, boxes have been classified in two classes depending on whether the pf^* values of their ancestors are bigger or smaller than their own pf^* values. Also, boxes containing a global minimizer point have been marked. Boxes containing minimizer points mostly have the largest pf^* values.

5 Numerical Results

The numerical experiments were carried out with a AMD K7 PC (500 MHz., 128 Mbyte RAM) running the Linux operating system. Programs were coded in C and C++. The inclusion functions were implemented via the PROFIL/BIAS routines (Ref. 14).

All derivatives are computed with forward mode automatic differentiation (see C++ Toolbox, Ref. 6). The standard time unit (the CPU time required to evaluate the Shekel 5 test function 1000 times at $(4.0, 4.0, 4.0, 4.0)^T$) was 0.0149 seconds.

In order to evaluate the capability of our algorithmic proposal for finding global minima, our computational experiments have been carried out using a wide set of test functions. This set of thirty four test problems, which includes several hard problems, is given in Table 1 where also numerical results for the execution of the basic algorithm are shown. Notice that for a set of six test function the basic algorithm did not finish after two hours running for $\epsilon = 10^{-6}$. This subset of very hard functions was also tested using the stopping criterion $\epsilon = 10^{-2}$ and again no solution was obtained after two hours. In Tables 1 to 3 the following notation is used for column headers:

Problem	The name of the test function and reference.
ϵ	The stopping criterion parameter ($w(X) < \epsilon$).
CPU_{1-C}	Execution time for Algorithm $1 - C$, $C = 2, 3$ (in seconds).

FX_{1-C} Number of interval evaluations of the function (as $F(X)$).

GX_{1-C} Number of interval evaluations of the derivative's function.

Fx_{1-C} Number of interval evaluations at x (as $F(x)$).

In Tables 2 and 3 results from the new algorithm are shown. The global minimum value for each function (f^*) was not provided to the algorithms.

The new algorithm was executed for the set of functions described in Table 1 using $\epsilon = 10^{-6}$ and $DecTh = 0.01$. This algorithm was able to solve all the problems except EX2, KW, N2, N3-10, R8 and RT20, for which it was stopped after two hours running. In these later cases the stopping criterion was too strong (ϵ too small), otherwise these problems are solvable by similar techniques (see e.g. the referred articles). For a subset of thirteen test functions the new algorithm was faster than the basic algorithm.

In Table 2, the computational cost of the new algorithm when solving this set of thirteen test functions is shown. The relative efficiency of the new algorithm compared to the basic algorithm is also provided in Table 2. The relative efficiency was defined as the ratio between the values of the performance indicators (CPU, FX, GX, Fx) obtained by the new algorithm and the basic algorithm, so the relative efficiency in term of the CPU performance indicator is defined as $\mathrm{CPU}_R = \mathrm{CPU}_{1-3}/\mathrm{CPU}_{1-2}$ and similarly for FX_R, GX_R and Fx_R. Hence lower values of the relative efficiency imply that the new algorithm is better than the basic algorithm. It can be seen that for this set of test functions, the new algorithm is approximately 35% faster than the basic algorithm. The best result was obtained for the Goldstein-Price function, for which the solution was by 96% cheaper computationally.

However, for a set of fifteen test functions the computational cost of the new algorithm was exactly the same as that of the basic algorithm. For these functions the new rejection criterion has no advantage because the values of pf^* were very small from the beginning of the algorithm; for some functions, values of pf^* even smaller than the value of $DecTh$ were obtained.

For the rest of 6 problems both algorithms used more than 2 hours of CPU time, and the results were slightly favorable for the new approach.

For the six hard problems (EX2, KW, N2, N3-10, R8 and RT20), the new algorithm was executed using other values for the stopping criterion ($\epsilon = 10^{-2}, 10^{-3}$, and 10^{-4}. Numerical results are shown in Table 3. The only two problems that the new algorithm could not solve for $\epsilon = 10^{-2}$ were RT20, which is defined in a large search domain ($[-600, 600]^{20}$), and the Neumaier 2 problem (N2). However, more accurate solutions were obtained for N3-10 and R8 ($\epsilon = 10^{-2}$) and ($\epsilon = 10^{-3}$), respectively. For EX2 and KW problems, the new algorithm spent around or less than half an hour when $\epsilon = 10^{-4}$ was used as stopping criterion.

Numerical results of Table 3 confirm that the new technique is capable to solve very hard problems, and that it should be incorporated in interval B&B algorithms for solving real life problems with high memory requirements. For hard to solve problems the new rejection test may make it possible to solve them, which are otherwise unsolvable with usual techniques. In these cases the new methodology can decrease the computational effort by as much as one order of magnitude.

The algorithm does not require the a priori knowledge of the global minimum, which can be estimated by the updated upper bound of the global minimum, $\tilde{f}$. Although all the global minima of the tested problems were found, according to our theoretical results, it cannot be proven for all possible cases. Further theoretical results indicate that the reliability of the improved algorithm can be proven for a wide class of inclusion functions. Another technique of saving the reliability can be the storing of the rejected sub-intervals into a file, and their later off-line processing.

For those problems where the accelerating devices does not help and a combinatorial explosion in the number of sub-boxes to be stored makes the B&B method intractable, the proposed algorithm is a viable solution. The new algorithm is also suitable for non-differentiable problems (Ref. 1) where advanced accelerating devices can not be applied. On the other hand, future tests may clear the efficiency changes caused by the new rejection criterion in sophisticated interval optimization algorithms with more accelerating devices. In a similar case (cf. Refs. 10–11, 15), the better interval subdivision direction selection technique proved to provide similar efficiency improvements in many quite different algorithm variants, with and without accelerating devices.

References

1. Casado, L. G., García, I., and Csendes, T., A Heuristic Rejection Criterion in Interval Global Optimization Algorithms, BIT, Vol. 41, No. 4, pp. 683–692, 2001.

2. Madsen, K. and Zertchanninov, S., A New Branch and Bound Method for Global Optimization, Technical Report IMM 05/98, Technical University of Denmark, Lyngby, Denmark, 1998.

3. Casado, L. G., García, I., and Martínez, J. A., Experiments with a New Selection Criterion in a Fast Interval Optimization Algorithm, Journal of Global Optimization, Vol. 19, pp. 247–264, 2001.

4. Ratschek, H. and Rokne, J., New Computer Methods for Global Optimization. Chichester, England: Ellis Horwood Ltd., 1988.

5. Ichida, K. and Fujii, Y., An Interval Arithmetic Method for Global Optimization, Computing, Vol. 23, pp. 85–97, 1979.

6. Hammer, R., Hocks, M., Kulisch, U., and Ratz, D., C++ Toolbox for Verified Computing I: Basic Numerical Problems: Theory, Algorithms, and Programs. Berlin, Germany: Springer-Verlag, 1995.

7. Hansen, E., Global Optimization Using Interval Analysis, Vol. 165 of Pure and Applied Mathematics. New York, NY: Marcel Dekker, Inc, 1992.

8. Kearfott, R. B., Rigorous Global Search: Continuous Problems. Dordrecht, Holland: Kluwer Academic Publishers, 1996.

9. Horst, R. and Pardalos, P., Editors., Handbook of Global Optimization, Vol. 2 of Noncovex Optimization and its Applications. Dordrecht, Holland: Kluwer Academic Publishers, 1995.

10. Ratz, D. and Csendes, T., On the Selection of Subdivision Directions in Interval Branch and Bound Methods for Global Optimization, Journal of Global Optimization, Vol. 7, pp. 183–207, 1995.

11. Berner, S., New Results in Verified Global Optimization, Computing, Vol. 57, No. 4, pp. 323–343, 1996.

12. Casado, L. G. and García, I., New Load Balancing Criterion for Parallel Interval Global Optimization Algorithms, in Proceedings of the 16th IASTED International Conference on Applied Informatics, (Garmisch-Partenkirchen, Germany), pp. 321–323, 1998.

13. Casado, L. G., García, I., and Csendes, T., A New Multisection Technique in Interval Methods for Global Optimization, Computing, Vol. 65, No. 3, pp. 263–269, 2000.

14. Knüppel, O., A Fast Interval Library, Computing, Vol. 53, pp. 277–287, 1994.

15. Csendes, T. and Ratz, D., Subdivision Direction Selection in Interval Methods for Global Optimization, SIAM Journal on Numerical Analysis, Vol. 34, pp. 922–938, 1997.

Table 1.

Symbol	Problem	Ref.	n	CPU_{1-2}	FX_{1-2}	GX_{1-2}	Fx_{1-2}
GP	Goldstein-Price	24	2	6.65	59,445	33,098	14,861
S5	Shekel 5	24	4	0.09	193	183	48
S7	Shekel 7	24	4	0.13	197	190	49
S10	Shekel 10	24	4	0.20	209	199	52
H3	Hartman 3	24	3	0.27	709	646	177
H6	Hartman 6	24	6	1.23	2,125	1,626	531
L3	Levy 3	25	2	2.32	2,401	1,908	600
L5	Levy 5	25	2	0.44	665	320	166
L8	Levy 8	25	3	0.05	151	128	38
SHCB	Six-Hump-Camel-Back	24	2	0.23	1,469	975	367
THCB	Three-Hump-Camel-Back	7	2	0.15	1,133	753	283

Continued from previous page

Symbol	Problem	Ref.	n	CPU_{1-2}	FX_{1-2}	GX_{1-2}	Fx_{1-2}
G2	Griewank 2	24	2	0.02	437	113	109
G10	Griewank 10	24	10	148.80	615,765	154,965	153,941
BR2	Branin 2	7	2	0.07	965	313	241
SRB2	Simplified Rosenbrock	7	2	0.02	409	116	102
RB2	Rosenbrock	7	2	0.02	409	119	102
RB10	Rosenbrock	20	10	152.97	704,189	177,853	176,047
P	Price	8	2	0.06	1,157	302	289
TR	Treccani	7	2	0.02	541	141	135
S3.1	Schwefel 3.1	25	3	0.03	183	157	49
MT	Matyas	25	2	0.03	757	197	189
EX1	EX1	6	2	0.02	109	89	27
EX2	EX2	6	5	>2h	—	—	—

Continued from previous page

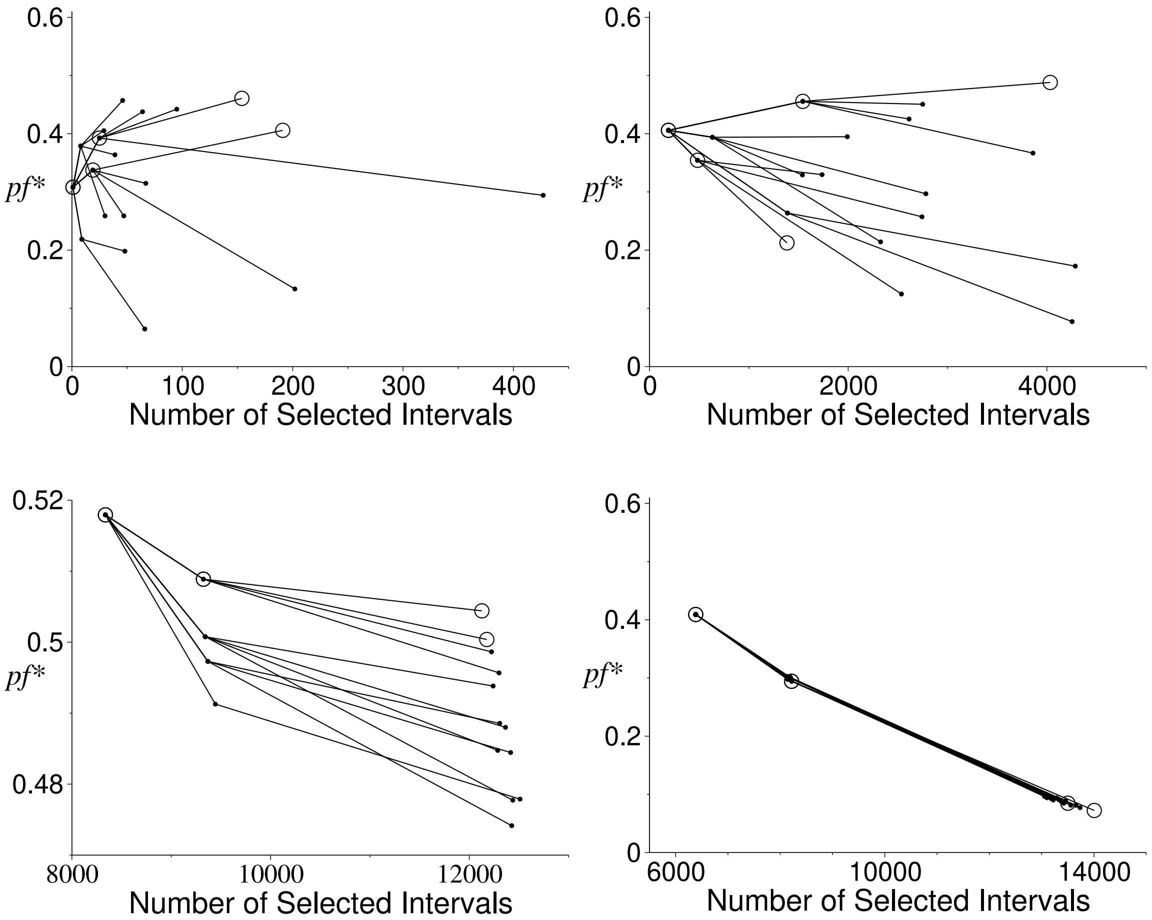
Symbol	Problem	Ref.	n	CPU_{1-2}	FX_{1-2}	GX_{1-2}	Fx_{1-2}
BR	Branin	24	2	0.06	1,033	366	258
KW	Kowalik	25	4	>2h	—	—	—
N2	Neumaier 2	20	4	>2h	—	—	—
N3-10	Neumaier 3	20	10	>2h	—	—	—
R4	Ratz 4	23	2	0.19	1,525	1,293	381
R8	Ratz 8	23	9	>2h	—	—	—
HM3	Henriksen-Madsen 3	11	2	0.85	3,445	1,897	861
HM4	Henriksen-Madsen 4	11	3	3.46	12,965	4,655	3,241
BL	Beale	25	2	0.06	745	310	186
CHI	Chichinadze	8	2	0.02	153	127	38
RT20	Rastringin	20	20	>2h	—	—	—

Table 2.

	$\epsilon = 10^{-6}$				Relative efficiency			
Prob.	CPU_{1-3}	FX_{1-3}	GX_{1-3}	Fx_{1-3}	CPU_R	FX_R	GX_R	Fx_R
GP	0.51	4,697	2,383	1,174	0.08	0.08	0.07	0.08
S5	0.09	193	178	48	1.00	1.00	0.97	1.00
S7	0.13	197	180	49	1.00	1.00	0.95	1.00
S10	0.19	209	188	52	0.95	1.00	0.94	1.00
H3	0.25	673	580	168	0.93	0.95	0.90	0.95
H6	0.95	1,629	1,240	407	0.77	0.77	0.76	0.77
L3	1.46	2,053	1,157	513	0.63	0.86	0.61	0.86
L5	0.28	445	198	111	0.64	0.67	0.62	0.67
SHCB	0.09	609	360	152	0.39	0.41	0.37	0.41
THCB	0.04	485	179	121	0.27	0.43	0.24	0.43
R4	0.15	1,309	1,040	327	0.79	0.86	0.80	0.86
HM3	0.51	2,269	1,118	567	0.60	0.66	0.59	0.66
HM4	0.83	3,517	1,050	879	0.24	0.27	0.23	0.27
				Average values	0.64	0.69	0.62	0.69

Table 3.

Problem	ϵ	CPU_{1-3}	FX_{1-3}	GX_{1-3}	Fx_{1-3}
EX2	10^{-2}	67.66	115,689	81,608	33,688
	10^{-3}	417.78	659,505	512,417	164,876
	10^{-4}	1,139.53	1,728,691	1,333,083	470,328
KW	10^{-2}	58.65	106,501	59,937	26,625
	10^{-3}	531.94	1,077,033	526,980	269,258
	10^{-4}	1,804.48	3,316,989	1,789,514	829,247
N2	10^{-2}	$>$ 2h	—	—	—
N3-10	10^{-2}	1,868.48	11,996,265	11,872,902	2,999,066
R8	10^{-2}	16.59	36,165	14,871	9,553
	10^{-3}	341.68	578,885	236,135	152,913
RT20	10^{-2}	$>$ 2h	—	—	—



0.6
0.4
pf*
0.2
0
0
100
200
300
400
Number of Selected Intervals
0.6
0.4
pf*
0.2
0
0
2000
4000
Number of Selected Intervals
0.52
0.5
pf*
0.48
8000
10000
12000
Number of Selected Intervals
0.6
0.4
pf*
0.2
0
6000
10000
14000
Number of Selected Intervals

Figure 1.

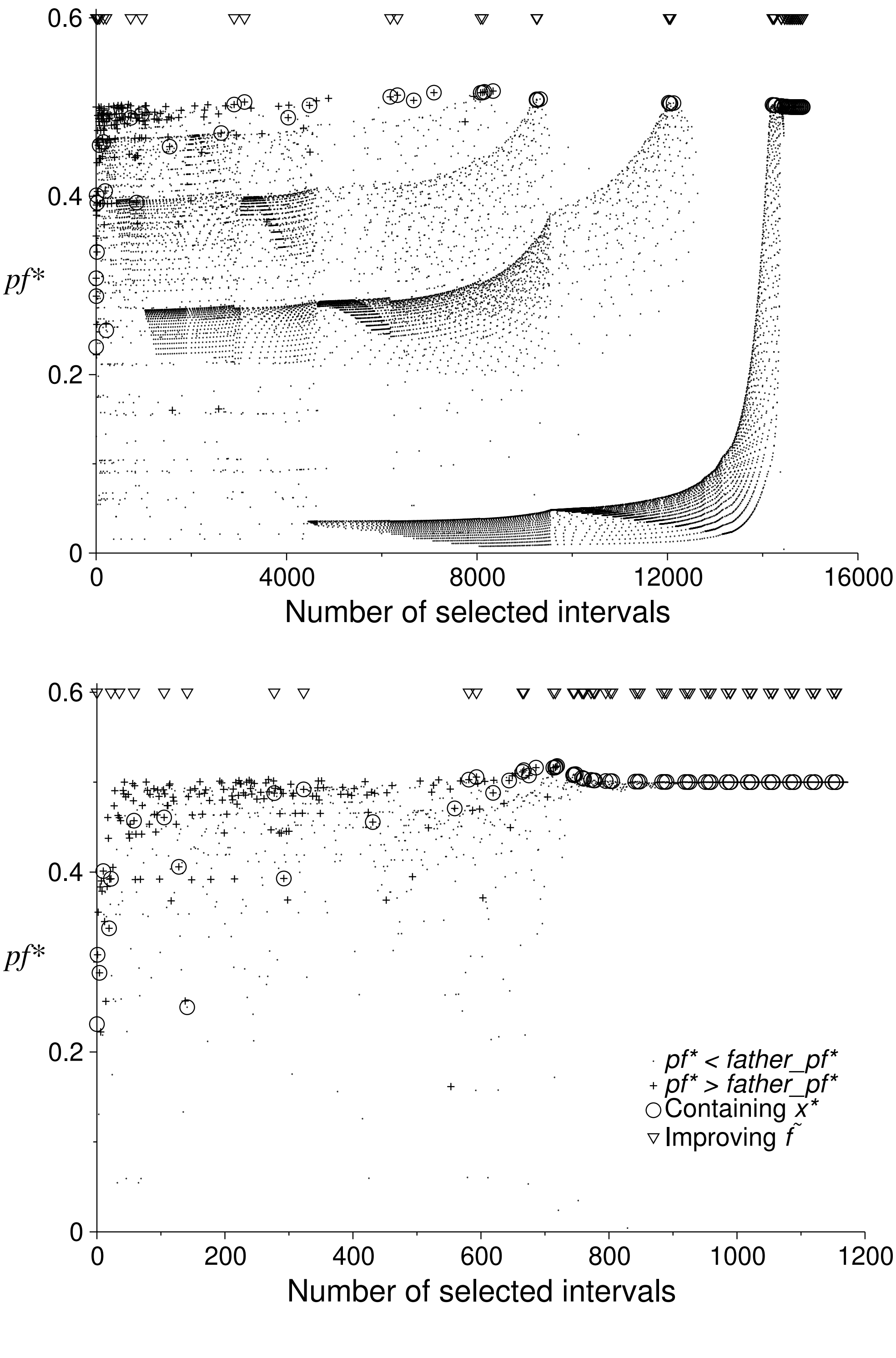

0.6
0.4
0.2
0
pf*
0
4000
8000
12000
16000
Number of selected intervals
0.6
0.4
0.2
0
pf*
0
200
400
600
800
1000
1200
Number of selected intervals
pf* < father_pf*
pf* > father_pf*
Containing x*
Improving f̃

Figure 2.

List of Tables

List of Figures