Journal of Global Optimization **19:** 247–264, 2001.
© 2001 *Kluwer Academic Publishers. Printed in the Netherlands.*

Experiments with a new selection criterion in a fast interval optimization algorithm [⋆]

L.G. CASADO, J.A. MARTíNEZ and I. GARCíA
Computer Architecture & Electronics Department, University of Almería Cta. Sacramento SN, 04120 Almería. Spain (e-mail: leo@ace.ual.es)

Abstract. Usually, interval global optimization algorithms use local search methods to obtain a good upper (lower) bound of the solution. These local methods are based on point evaluations. This paper investigates a new local search method based on interval analysis information and on a new selection criterion to direct the search. When this new method is used alone, the guarantee to obtain a global solution is lost. To maintain this guarantee, the new local search method can be incorporated to a standard interval GO algorithm, not only to find a good upper bound of the solution, but also to simultaneously carry out part of the work of the interval B&B algorithm. Moreover, the new method permits improvement of the guaranteed upper bound of the solution with the memory requirements established by the user. Thus, the user can avoid the possible memory problems arising in interval GO algorithms, mainly when derivative information is not used. The chance of reaching the global solution with this algorithm may depend on the established memory limitations. The algorithm has been evaluated numerically using a wide set of test functions which includes easy and hard problems. The numerical results show that it is possible to obtain accurate solutions for all the easy functions and also for the investigated hard problems.

Key words: Interval arithmetic, Branch & bound, Global optimization, Selection rules

1. Introduction

Interval global optimization is a research field which investigates the problem of finding the set of optimizer points in a given domain using interval methods. The state of the art in this subject along with the increasing computing power allow the solution of many global optimization problems. Nevertheless, even using all the available accelerating devices, there exists a set of hard problems for which not even the most powerful computers are able to find accurate solutions. In fact, the main reasons why an Interval Global Optimization (IGO) algorithm does not reach the solution of a specific instance are: (a) large overestimations of the inclusion function and its derivatives, (b) the instance to be solved contains many local minimizer points, (c) the instance is of high dimension, (d) the corresponding objective function is not differentiable. Often, the amount of memory required in order to solve this kind of problems is higher than the computer's memory resources.

[⋆] This work was supported by the Ministry of Education of Spain (CICYT TIC99-0361).

We are interested in a fast and simple interval B&B algorithm. In this sense, the Moore-Skelboe algorithm [18] with the midpoint test [12] is one of the simplest B&B algorithms suitable for minimizing differentiable and non differentiable functions. However the Moore-Skelboe algorithm performs badly when difficult problems are treated and/or very accurate solutions are required. This is due to the fact that the stored n-dimensional interval vectors (boxes) in the work list can run out of the available random access memory, slowing down the execution of the algorithm or hindering the attainment of very accurate solutions. The investigated algorithm is a variant of the Moore-Skelboe algorithm that explicitly uses a limited budget for the computer memory resources. This budget is a parameter of the algorithm which controls the computational effort applied to the problem at hand and the maximum number of boxes in the final list.

In the next sections, after defining the general Interval Branch and Bound framework, a discussion about the subdivision, selection and elimination criteria is made. Section 4 is devoted to briefly describing our algorithm which intends to find a fast solution for the global optimization problem. In Section 5 the algorithm is evaluated by a set of experimental results obtained from the execution of the algorithm using a wide set of test functions.

2. Definitions and General Framework

This paper investigates a variant of an Interval Branch-and-Bound (B&B) algorithm [18] for solving the bound constrained global optimization problem [9, 10, 22]:

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

where the n-dimensional interval $S \subseteq \mathbb{R}^n$ is the search region, and $f : \mathbb{R}^n \to \mathbb{R}$ is continuous.

The global minimum value of f is denoted by f^* and the set of global minimizer points of f on S by X^*. That is,

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

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 $\overline{X}_i = \max\{x \in X_i\}$, for $i = 1, 2, \ldots, n$. The set of compact intervals is denoted by $\mathbb{I} := \{[a, b] \mid a \leqslant b,\ a, b \in \mathbb{R}\}$ and the set of n-dimensional interval vectors (also called boxes) by $\mathbb{I}^n$.

For real and interval vectors the notation

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

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

The general framework for B&B algorithms can be characterized by the following five rules: bounding, branching, termination, elimination and selection. They can be briefly described as follows:

The bounding rule is given by interval arithmetic [16] which allow us to obtain an *inclusion function*. A function $F : \mathbb{I}^n \to \mathbb{I}$ is called an *inclusion function* of f in $X \subseteq \mathbb{R}^n$, when $x \in X$ implies $f(x) \in F(X)$. In the present study the inclusion function of the objective function is obtained by natural interval extension [18] (real operations and standard functions are simply substituted by their interval equivalents).

The branching rule used here consists of bisecting the two widest sides of the current box in one step. More than two bisections in one step are not used to avoid the explosion of the number of generated boxes. Other branching rules, some of them using derivative information, can be found in [1, 7, 16, 19, 20].

The termination rule is generally based on the values of $w(X)$ and/or $w(F(X))$ [18]. In the present case, $w(X) < \epsilon$ was required as a stopping criterion for all the remaining boxes.

The simplest and basic elimination rules are the midpoint and cut-off tests, where boxes with $f^* \leqslant \overline{f^*} < \underline{F}(X)$ are discarded ($\overline{f^*}$ is the current upper bound of the minimum, updated by evaluating $\overline{F}(m(X))$). In addition to the midpoint and cut-off tests, the monotonicity test, the non-convexity test [7] and the interval Newton method [7, 18] can be applied. It is usually worth to use accelerating devices based on derivative information because the number of evaluated boxes can be decreased but the computational effort applied to a single box increases as the dimension of the objective function increases. These tests might be less useful when the objective function is multimodal and the related inclusion functions overestimate the real range too much.

The usual selection rule consists of choosing, from the set of generated and non rejected boxes (current set), that one with the smallest value of $\underline{F}(X)$, using a Best-First search strategy. With the Best-First search strategy, boxes in the current set are located on different levels of the search tree and they differ in size and shape. For α-convergent inclusion functions [18], the overestimations of the range obtained by the inclusion function depend (but not only) on the width of the given interval. Therefore, a box can be selected because it is wider than the others, and not because it contains a global minimizer point. As an example of this behaviour, the first iterations of the Moore-Skelboe algorithm executed on the function $f(x) = \sum_{i=0}^{3}(-1)^i \sin(2\pi i x)$ has been drawn in Figure 1.

In Figure 1 each box represents an interval $[\underline{X}, \overline{X}]$ and its upper and lower bounds $[\underline{F}(X), \overline{F}(X)]$. The whole search region X^1 is selected and divided using bisection, generating subintervals X^2 and X^3 which are evaluated by the inclusion function. It can be observed that $\underline{F}(X^3) = \underline{F}(X^1)$ and $\overline{F}(X^2) = \overline{F}(X^1)$. The next selected interval (from the current set of boxes: $\{X^3, X^2\}$) to be divided is

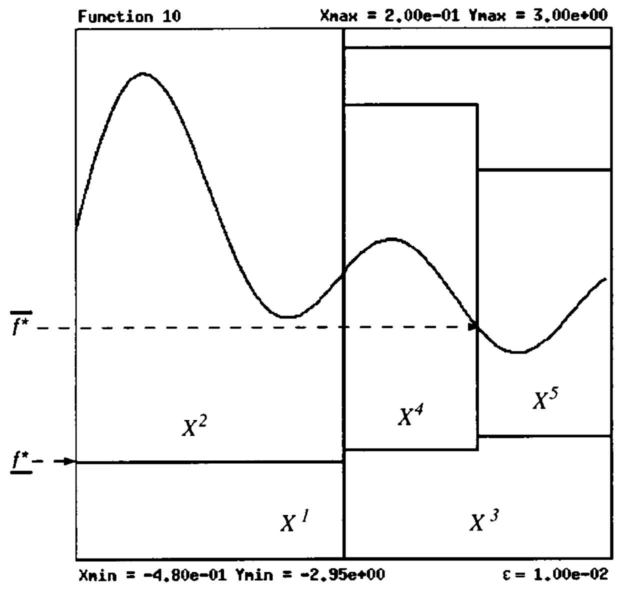

Figure 1. First iterations of the Moore-Skelboe algorithm on the problem of $f(x) = \sum_{i=0}^{3}(-1)^i \sin(2\pi i x)$.

X^3, because it has the smallest lower bound, generating subintervals X^4 and X^5, which are evaluated too. At this point the current set consists of $\{X^2, X^4, X^5\}$. Following the selection rule, the next processed interval will be X^2, although it does not contain the global minimum. It can be observed that even for intervals with the same width (X^4 and X^5) the overestimation can be significantly different, but these differences are smaller than those obtained between intervals with a different width. The next section is devoted to the analysis of the advantages and disadvantages of using an alternative selection criterion.

3. Discussion

An alternative selection criterion to the Best-First based on $\underline{F}(X)$ may consist of choosing from the current set that box with the largest value of a parameter $p\overline{f^*}(X)$ which was defined in [2] as:

$$p\overline{f^*}(X^i) = \frac{\overline{f^*} - \underline{F}(X^i)}{w(F(X^i))} \in [0, 1], \quad \forall X^i \in S.$$

The numerator is equivalent to the $\underline{F}(X)$ selection criterion, because the value of $\overline{f^*}$ is the same for every box in a fixed iteration. However, the numerator is normalized by $w(F(X))$. Thus, the parameter $p\overline{f^*}(X)$ is also based on the value of $\overline{F}(X)$. In the framework of a B&B algorithm, using this new parameter, any box X with $p\overline{f^*}(X) < 0$ will be eliminated by the cut-off test; so any box at the current set of boxes always verifies that $0 \leqslant p\overline{f^*}(X) \leqslant 1$. In [2] $p\overline{f^*}$ has been successfully used for solving the work load balancing problem in a parallel computing environment, where it was used as an estimator of the computational work load of any box.

Let us now show some experimental results which provide valuable information about the use of $\underline{F}(X)$ or $p\overline{f^*}(X)$ as a parameter of the selection rule. As an example, Figure 2 shows a comparison between both selection rules: Best-First based on $\underline{F}(X)$ (upper graph) and Best-First based on $p\overline{f^*}$ (bottom graph), for a standard test function (Levy 3). Both graphs represent the values of $p\overline{f^*}$ for all the selected boxes as the algorithm runs (rejected boxes are not represented in these graphs because their $p\overline{f^*}$ value is negative). During the algorithm execution, boxes containing a global minimizer point have been shown as big circles. In these graphs, a triangle has been drawn when the evaluation of $\overline{F}(m(X))$ produced an improvement for the value of $\overline{f^*}$. The best selection rule should choose from the current set of boxes the one which contains a global minimizer point because it increases the chances of obtaining good values of the upper bound of f^*.

Using a Best-First $\underline{F}$ selection rule, hardly any box is unnecessarily subdivided [1], but the memory requirements exponentially depend on the width and the depth of the search tree and the solution (minimum value of $\overline{f^*}$) is usually found at the final or intermediate stages of the algorithm execution (see upper graph of Figure 2).

On the other hand, the Best-First $p\overline{f^*}$ selection rule works like a Depth-First search. The advantage of a Depth-First search is its linear memory complexity and quick improvement of $\overline{f^*}$. The graph at the bottom of Figure 2 shows that all of the $\overline{f^*}$ improvements are done at the beginning of the algorithm execution because it converges very quickly to one global minimizer point (the Levy 3 problem has nine global minimizer points). The main drawback of a Depth-Search method is that it produces a sequence of local searches and after reaching a local or global minimizer point, it usually evaluates many boxes located in the region of attraction to that minimizer point, so it can take a long time to leave the region of attraction to a global or a local solution before it looks for another solution. Examples of this behaviour are shown in the bottom graph of Figure 2 and in Figure 3.

An alternative to these search strategies may consist in mixing local and global searches. It means that after a depth search which cannot go on to the next level of the search tree, a new branch is selected from the highest levels of the search tree to start a new depth search, and so on. A depth search cannot continue downwards when the boxes generated by a subdivision are rejected or they belong to the solution set. Several selection rules based on this kind of mixed searches (named hybrid selection rules) have been analysed in [11, 21]. An example of the execution of a B&B algorithm which applies a hybrid selection rule based on the value of $p\overline{f^*}$ is shown in Figure 4, where the best value of $\overline{f^*}$ is found in the early stages of the algorithm execution [2]. Also, all the sequences of selected boxes which

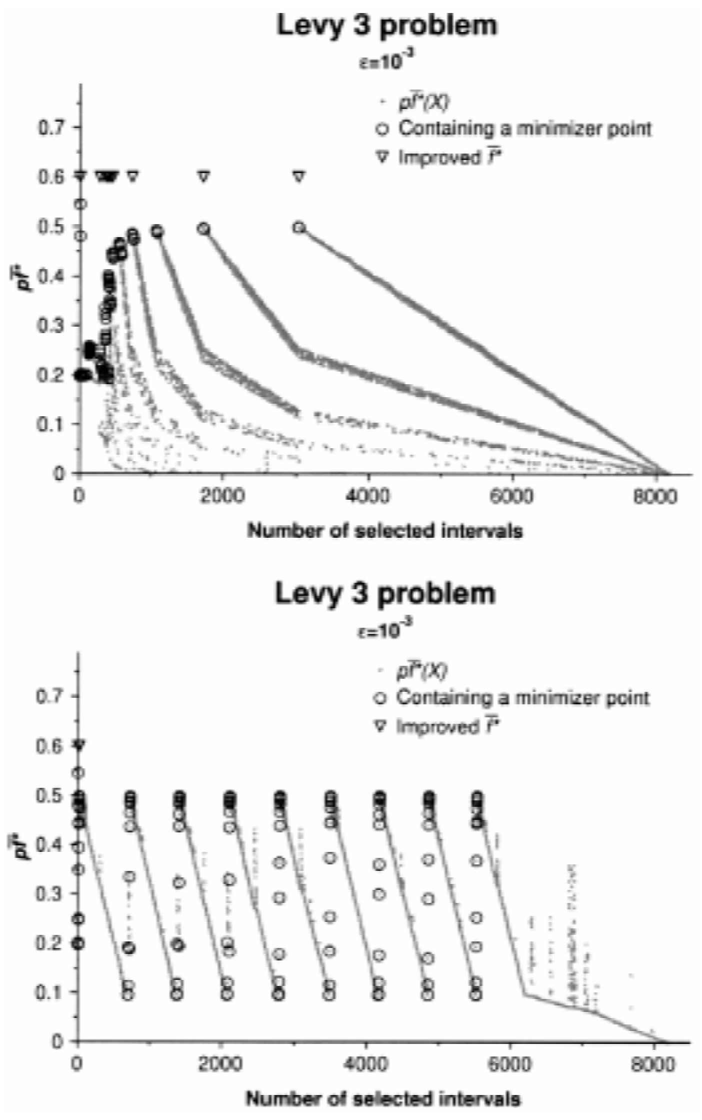

Figure 2. Values of $p\overline{f}^*$ for the Levy 3 function with $\epsilon = 10^{-3}$, using $\underline{F}(X)$ (upper graph) and $p\overline{f}^*$ (bottom graph) in a Best-First selection criterion.

converge to the global minimum are examined at the first iterations of the algorithm execution. The graph in the bottom of Figure 4 is a zoom of the upper graph.

Let us now suppose that the global optimization problem to be solved is a very hard problem and after running a B&B algorithm for several hours it was impossible to obtain a solution. In this situation, it can be decided to stop the algorithm execution after a specified number of iterations or when the computer runs out of memory, obtaining in this way an approximate or the true solution of

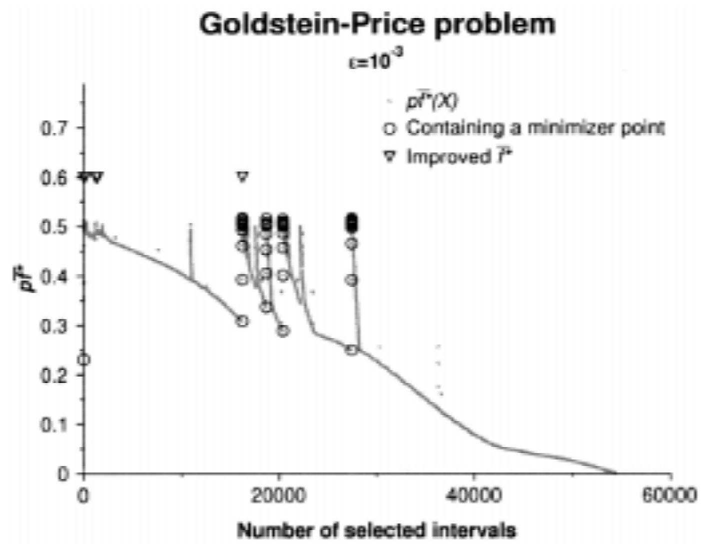

Figure 3. Values of $p\overline{f^*}$ for the Goldstein-Price function for $\epsilon = 10^{-3}$, using the Best-First $p\overline{f^*}$ selection criterion.

the problem. For these cases, the obtained solution may depend on the selection criterion used by the B&B algorithm. The previous experiments have shown that using a hybrid selection criterion the chance of finding the global (or a better) solution at the first stage of the algorithm execution is higher than using a Best-First strategy based on $\underline{F}(X)$.

In the next section a fast interval B&B algorithm based on a new hybrid selection rule is described. Section 5 is devoted to evaluating our algorithmic proposal and to comparing it with a traditional B&B algorithm using a Best First selection rule based on $\underline{F}(X)$.

4. Algorithmic Description

Globally, the algorithm we are investigating in this work follows a traditional B&B framework and consists of two stages. The goal of the first stage is to improve the value of $\overline{f^*}$ as much as possible by a fast local search which incorporates a special hybrid selection criterion. The second stage intends to assure a guaranteed global solution and works on the set of boxes which were generated (but not processed) during the first stage. Our main interest is to find out if the algorithm is able to reach the global solution during the first stage. For the second stage any of the traditional B&B algorithms can be used, so it is not described here (another approach could consist of repeating the first phase on the set of non processed boxes).

This first stage is done by the FIO (Fast Interval Optimization) algorithm, described by Algorithm 1. For the description of FIO algorithm, the following notation has been used:

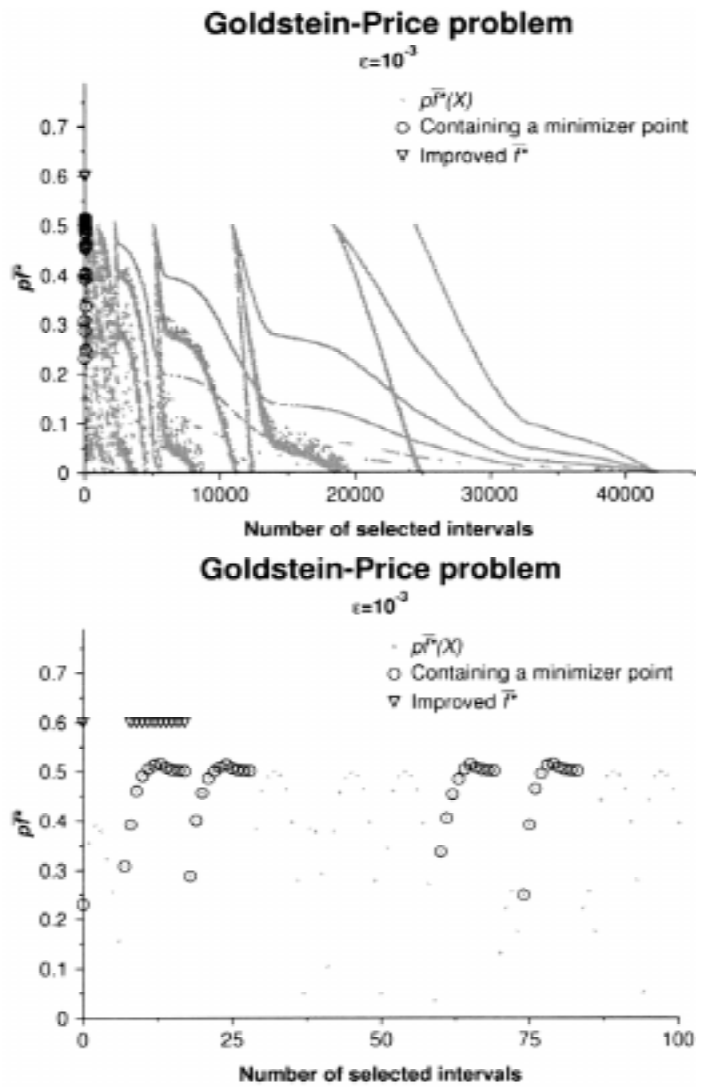

Figure 4. Values of $p\overline{f^*}$ for the Goldstein-Price problem with $\epsilon = 10^{-3}$, using a hybrid selection rule. Bottom graph is a zoom of the upper one.

S:	Search region.
F:	Natural interval extension of function f.
ϵ:	Stopping criterion parameter ($w(X) < \epsilon$).
L_0:	Work list.
L_1:	Temporary list.
L_2:	Secondary list.
Q:	Final list.
N_m:	Maximum number of boxes to be processed at each level of the search tree.
$+/-$:	Denote including/discarding boxes in a list.

ALGORITHM 1. Fast interval optimization algorithm

```
**funct FIO** $(S, F, \epsilon, L_2, Q, N_m)$
 $\overline{f^*} = \overline{F}(S)$                                   *Upper bound of $f^*$*
 $L_0 := \{S\}$                                                      *Work list*
 $L_1 := \{\}$                                                       *Temporary list*
 $L_2 := \{\}$                                                       *Secondary list*
 $Q := \{\}$                                                         *Final list*
 WHILE ( $L_0 \neq \{\}$ )
     NSelect:=0                                   *Number of selected boxes from $L_0$*
     WHILE ( $L_0 \neq \{\}$ AND NSelect $< N_m$)
         $X$:=Head($L_0$)                   $p\overline{f^*}(X) = \max\{p\overline{f^*}(X^i),\ \forall X^i \in L_0\}$
         $L_0 := L_0 - \{X\}$
         NSelect:=NSelect+1
         $\overline{f^*} := \min\{\overline{f^*}, \overline{F}(m(X))\}$            *Updating $\overline{f^*}$*
         IF ($\overline{f^*} = \overline{F}(m(X))$)                      *$\overline{f^*}$ just improved*
             $L_0$:=CutOffTest($L_0, \overline{f^*}$)
                        Comment: $\forall X^i \in L_0,\ \overline{f^*} < \underline{F}(X^i) \Rightarrow L_0 := L_0 - \{X^i\}$
             $L_1$:=CutOffTest($L_1, \overline{f^*}$)
             $L_2$:=CutOffTest($L_2, \overline{f^*}$)
             $Q$:=CutOffTest($Q, \overline{f^*}$)
         Subdivide $X$ into $X^1, ..., X^4$
         FOR i:=1 TO 4
             IF ( $\overline{f^*} < \underline{F}(X^i)$ NEXT i                    *Midpoint test*
             IF ( $w(X^i) < \epsilon$ ) $Q := Q + \{X^i\}$                   *Save $X^i$ in $Q$*
             ELSE $L_1 := L_1 + \{X^i\}$                                *Save $X^i$ in $L_1$*
     $L_2 := L_2 + L_0$
     $L_0 := L_1$                                                 *Reject boxes from $L_0$*
     $L_1 := \{\}$
End pseudo-code
```

The main differences between the FIO algorithm and the Moore-Skelboe algorithm are the following:

– The selection criterion is based on the $p\overline{f^*}$ value; the N_m boxes with the greatest $p\overline{f^*}$ values will be selected and processed at every level of the search tree.
– The FIO algorithm uses four different lists: A work list (L_0) which stores the set of candidate boxes to be selected for subdivision (the maximum number of selected boxes from L_0 is N_m, a user given parameter); a temporary list (L_1) where the boxes obtained by subdividing the selected boxes from the work list L_0 are saved (the maximum number of boxes in L_1 and L_0 is $N_m \times 2^d$, where 2^d is the number of subintervals obtained after bisecting the d widest dimensions of a box. In our implementations $d = 2$); a secondary list (L_2) with the set of boxes in L_0 which were not selected for subdivision; i.e. boxes which will be processed in the second stage of the B&B algorithm (the maximum number of boxes at L_2 is $N_m \times (2^d - 1) \times l_{max}$, where l_{max} is the maximum

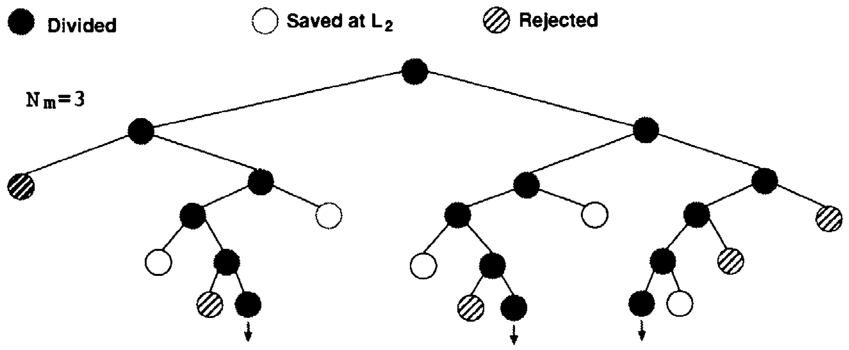

Figure 5. An example of the search tree produced by the FIO algorithm with $N_m = 3$ and with a bisection subdivision strategy.

level of the search tree to reach the stopping criterion); a final list Q containing the set of boxes which reached the termination criterion ($w(X) \leqslant \epsilon$).

The set of boxes in L_0 are ordered by a non-decreasing $p\overline{f^*}(X)$ value. Those boxes with the same $p\overline{f^*}$ value are ordered by a non-decreasing $\overline{F}(X)$ value and those with the same $\overline{F}(X)$ value are ordered by the Oldest-First criterion. In the implementation of Algorithm 1, instead of simple linked lists, balanced binary trees are applied, because they improve the complexity of the ordered insertion. In these trees, each node is a linked list containing boxes with the same $p\overline{f^*}$ value.

Algorithm 1 starts by initiating the work list to the search region S. The final, secondary and temporary lists are empty at the beginning of the algorithm. The algorithm consists of two loops: the outer loop runs for every level of the search tree and the inner loop works on boxes belonging to the same level of the search tree. The inner loop runs while the work list is not empty and the number of processed boxes at the current level of the search tree is less than N_m. All the boxes not selected at this level of the search tree are moved to the secondary list. Boxes obtained from the subdivision of the N_m selected boxes are stored in the temporary list L_1. At the next level of the search tree the work list is initiated with the set of boxes saved at the temporary list. Whenever the value of $\overline{f^*}$ is improved, the CutOffTest is applied not only to the work and final lists but also to the temporary and secondary lists. The FIO algorithm returns the final list with boxes X ($w(X) \leqslant \epsilon$) which most probably contain the global solution, and the secondary list with the set of non selected boxes.

The selection criterion used in the FIO algorithm tries to make global decisions about the next box to be processed because all the candidates belong to the same level of the search tree. This also ensures a better comparison between them because all the candidates have the same width and shape. This scheme follows a Breadth-First selection strategy which usually needs high memory requirements, but in the FIO algorithm the memory requirements are fixed by N_m. This threshold,

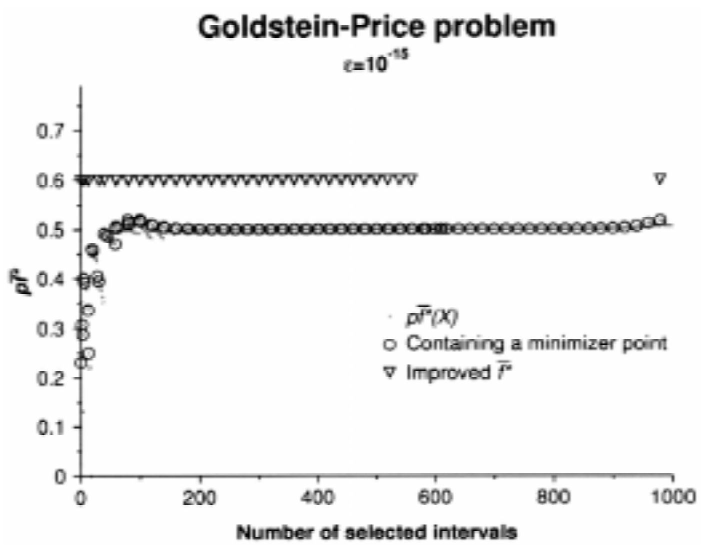

Figure 6. Values of $p\overline{f^*}(X)$ for boxes selected during the execution of Algorithm 1 with N_m = 20.

must depend on the available memory and computational resources. It is evident that the greater the value of N_m, the wider the search region where the algorithm works and the greater the probability of convergence to the global solution. A graphical example of the search tree for $N_m = 3$ and a bisection subdivision strategy ($d = 1$) is shown in Figure 5.

As an example, using the Goldstein-Price function, Figure 6 graphically shows the values of $p\overline{f^*}(X)$ for those boxes X which were selected during the execution of the FIO algorithm. In this example, the termination criterion was $w(X) \leqslant \epsilon = 10^{-15}$, which is near to the minimum value that can be achieved with a Pentium-II processor due to the applied rounding. In this example, for which heavy restrictions for memory requirements were established ($N_m = 20$), the problem was solved ($[\underline{f^*}, \overline{f^*}]$ = [2.999999999999257, 3.000000000000050]) with less than 5,000 interval function evaluations and 0.15 seconds of CPU time. The number of boxes in the final list was $80 = N_m \times 2^d$ and one of them contained the global solution.

5. Numerical Results

This section intends to show experimental results attempting to determine under which conditions the FIO algorithm is able to obtain the global solution for a GO problem. These numerical results can also be seen as a measurement of the capability of the algorithm for being used as a good local search procedure.

The experiments were carried out with a Pentium-II PC (233 Mhz., 256 Mbyte RAM) running the Linux operating system. Programs were coded in C and C++. The inclusion functions were implemented via the PROFIL/BIAS routines [13, 14, 15]. All derivatives are computed with automatic differentiation (see C++ Toolbox [6]), where the usual techniques of the forward mode of automatic differentiation have been applied. 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 0.0215 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 described in Table 1 where also numerical results for the Moore-Skelboe algorithm with the standard Best-First $\underline{F}$ selection criterion, midpoint, cut-off and monotonicity tests as the only accelerating devices are shown. The division rule used to obtain the results in Tables 1 to 3, consisted in bisecting the two widest dimensions of a box in one step and the stopping criterion was always $w(X) \leqslant \epsilon = 10^{-6}$.

In the next tables the following notation has been used for column headers:

Prob.:	The name of the test function.
Ref.:	Reference where the problem is described.
n:	Dimension of the problem.
N_m:	Maximum Number of boxes to be processed at each level of the search tree.
CPU:	Execution time in seconds.
FX:	Number of $F(X)$ evaluations.
GX:	Number of $\nabla F(X)$ evaluations.
Fx:	Number of $F(m(X))$ evaluations. It is equal to the number of boxes extracted from the work list.

Results in Table 1 show that for most of the functions the Moore-Skelboe algorithm is able to find the global solution very fast, however for a set of six test functions (EX2, KW, N2, N3-10, R8 and RT20) the program was stopped after two hours.

Table 2 shows the computational cost of the FIO algorithm evaluated with $p\overline{f^*}$ and $\underline{F}$ as the parameter used by the selection criterion (line 10 of Algorithm 1). Notice that the only difference between both alternatives is the criterion applied to sorting the list L_0; a non-increasing order of the value of $p\overline{f^*}$ and a non-decreasing order of the value of $\underline{F}$. Results in Table 3 were obtained by adding the monotonicity test to the FIO algorithm. In Tables 2 and 3, the values of the CPU time and the number of interval evaluations (FX, GX and Fx) from executions of the FIO algorithm were computed using a specific value of N_m. For each function the value of N_m was chosen in such a way that at least one of the global minima of the function was contained in the set of boxes of the final list Q; i.e. the minimum value of N_m for which the global solution was found. That is why N_m differs from problem to problem.

From a computational perspective, in Table 2 it can be seen that the FIO algorithm with a selection criterion based on $\underline{F}$ is more expensive than that based on

Table 1. Data on the set of thirty four test functions and numerical results for the Moore-Skelboe algorithm with the standard Best-First $\underline{F}$ selection criterion, midpoint, cut-off and monotonicity tests, $\epsilon = 10^{-6}$ and using bisection of the two widest dimensions of a box in one step as division rule

Prob.		Ref.	n	CPU	FX	GX	Fx
GP	Goldstein-Price	[22]	2	12.62	59,445	33,098	14,861
S10	Shekel 10	[22]	4	0.37	209	199	52
S7	Shekel 7	[22]	4	0.25	197	190	49
S5	Shekel 5	[22]	4	0.18	193	183	48
H6	Hartman 6	[22]	6	2.48	2,125	1,626	531
H3	Hartman 3	[22]	3	0.52	709	646	177
L3	Levy No. 3	[23]	2	5.47	2,401	1,908	600
L5	Levy No. 5	[23]	2	1.07	665	320	166
L8	Levy No. 8	[23]	3	0.09	153	130	38
C	Six Hump Camel Back	[22]	2	0.42	1,469	975	367
C3	Three Hump Camel Back	[4]	2	0.28	1,133	753	283
G2	Griewank 2	[22]	2	0.05	437	113	10
G10	Griewank 10	[22]	10	380.73	615,765	154,965	153,941
BR2	Branin 2	[4]	2	0.14	965	313	241
SRB2	Simplified Rosenbrock	[4]	2	0.03	409	119	102
RB2	Rosenbrock	[4]	2	0.03	409	116	102
RB10	Rosenbrock	[17]	10	47.27	118,897	30,537	29,724
P	Price	[5]	2	0.11	1,157	302	289
TR	Treccani	[4]	2	0.03	541	141	135
S3.1	Schwefel No. 3.1	[23]	3	0.05	197	165	49
MT	Matyas	[23]	2	0.05	757	197	189
EX1	EX1	[3]	2	0.03	109	89	27
EX2	EX2	[3]	5	>2h	—	—	—
BR	Branin	[22]	2	0.12	1,033	366	258
KW	Kowalik	[23]	4	>2h	—	—	—
N2	Neumaier 2	[17]	4	>2h	—	—	—
N3-10	Neumaier 3	[17]	10	>2h	—	—	—
R4	Ratz No. 4	[20]	2	0.39	1,525	1,293	381
R8	Ratz No. 8	[20]	9	>2h	—	—	—
HM3	Henriksen-Madsen No. 3	[8]	2	2.04	3,445	1,897	861
HM4	Henriksen-Madsen No. 4	[8]	3	8.89	12,965	4,655	3,241
BL	Beale	[23]	2	0.11	745	310	186
CHI	Chichinadze	[5]	2	0.05	153	127	38
RT20	Rastringin	[17]	20	>2h	—	—	—

$p\overline{f^*}$ for a wide set of functions (the following sixteen functions: GP, S10, S7, S5, H6, H3, L3, L5, C, C3, EX1, EX2, KW, N3-10, R4, and CHI). Only for functions S3.1, HM3, HM4 and BL the FIO algorithm based on $\underline{F}$ behaves better than that based on $p\overline{f^*}$. For the remaining fourteen functions, similar numerical results were obtained for both parameters ($\underline{F}$, $p\overline{f^*}$). However, using the monotonicity test (see Table 3), the results obtained with $p\overline{f^*}$ are actually better than those of $\underline{F}$ only for functions GP, H3, L3, L5, C, C3, EX2, KW, N3-10 and R4, while for functions S3.1, HM3, HM4 and BL the use of $\underline{F}$ is less expensive than $p\overline{f^*}$. For the remaining twenty test functions both parameters produce similar numerical results. From these results it can be said that for deciding which box contains a global minimizer point the parameter $p\overline{f^*}$ is a better estimator than the parameter $\underline{F}$. Notice also that for function N3-10 the FIO algorithm with $\underline{F}$ did not find the global solution after six hours even using the monotonicity test.

The numerical results in Tables 2 and 3 show that for a wide set of test functions it is possible to select a small value of N_m for which at least one global minimizer point is found and also that for extremely hard functions the FIO algorithm using $p\overline{f^*}$ was able to find a global minimizer point very fast, even without applying the monotonicity test. For all the functions, the value of N_m is less or equal when monotonicity test is used. However, the CPU time for the FIO algorithm is greater with than without using the monotonicity test because the algorithm has to evaluate the derivative of the inclusion function for most of the processed boxes.

In general, the fastest solutions for the FIO algorithm were obtained with a selection criterion based on $p\overline{f^*}$ and without monotonicity test. Thus, the FIO algorithm is also suitable for non-differentiable problems where advanced accelerating devices cannot be applied.

It is worth mentioning that for many functions the minimum value of N_m ($N_m = 1$) was sufficient to find a global minimizer point. One of the functions included in this set is RT20 which was not solved by the Moore-Skelboe algorithm.

Due to the heuristic characteristic of the parameters used in the selection rule ($p\overline{f^*}$, $\underline{F}$) the minimum value of N_m, with which a global minimizer point can be found, is not known in advance. One of the reasons why N_m should be greater than one is that two or more boxes at the same level of the search tree have the same value of the parameter ($p\overline{f^*}$ or $\underline{F}$).

Although numerical results in the Tables have shown that the FIO algorithm has found a global minimizer point for the set of tested functions, there are problems where this local search algorithm does not converge to a global minimizer point with the available computational resources. An example of these problems is the fifteen dimensional case of the N3 problem (N3-15) [17].

6. Concluding Remarks

In this work an interval B&B algorithm which incorporates a new selection criterion has been analyzed. Its ability to obtain very fast improvements of $\overline{f^*}$ has

Table 2. Numerical results of the FIO algorithm with $\epsilon = 10^{-6}$ and using bisection of the two widest dimensions of a box in one step as division rule

	Selection Criterion based on $p\overline{f^*}$				Selection Criterion based on $\underline{F}$			
Prob.	CPU	N_m	FX	Fx	CPU	N_m	FX	Fx
GP	0.01	2	173	43	17.15	11,140	518,317	129,579
S10	0.04	2	369	92	0.54	43	5,785	1,446
S7	0.07	5	881	220	0.31	29	4,189	1,047
S5	0.11	11	1,725	431	0.19	24	3,273	818
H6	0.70	31	7,153	1,788	46.83	2,315	452,693	113,173
H3	0.08	10	1,125	281	2.50	375	35,293	8,823
L3	0.02	1	101	25	0.04	2	197	49
L5	0.02	1	101	25	0.08	4	377	94
L8	0.02	1	153	38	0.02	1	153	38
C	0.01	3	281	70	0.80	409	27,253	6,813
C3	0.20	341	7,689	1,922	0.38	1,387	17,637	4,409
G2	0.01	1	113	28	0.01	1	113	28
G10	0.14	1	621	155	0.15	1	621	155
BR2	0.02	4	373	93	0.02	4	385	96
SRB2	0.01	1	89	22	0.01	1	89	22
RB2	0.01	1	89	22	0.01	1	89	22
RB10	0.03	1	441	110	0.03	1	441	110
P	0.01	1	101	25	0.01	1	101	25
TR	0.01	1	97	24	0.01	1	97	24
S3.1	0.02	5	629	157	0.01	1	153	38
MT	0.01	1	101	25	0.01	1	101	25
EX1	0.01	4	317	79	0.03	8	581	145
EX2	11.71	529	91,873	22,968	370.13	17,973	2,764,613	691,153
BR	0.01	2	189	47	0.01	2	189	47
KW	1.94	155	21,069	5,267	72.35	6,708	783,293	195,823
N2	5.51	740	53,821	13,455	5.70	740	53,821	13,455
N3-10	31.94	965	522,465	130,616	—	$>10^5$	—	—
R4	0.05	10	861	215	0.13	29	2,401	600
R8	0.63	5	2,233	558	0.62	5	2,241	560
HM3	1.21	109	7,461	1,865	0.02	1	101	25
HM4	2.41	96	12,001	3,000	1.64	64	8,509	2,127
BL	0.02	9	781	195	0.01	6	549	137
CHI	0.01	1	105	26	0.63	119	10,213	2,553
RT20	0.44	1	1,241	310	0.44	1	1,241	310

been experimentally demonstrated. The algorithm can be seen from two perspectives: (a) As a local optimizer based on interval evaluations which can be useful as a first stage of any interval global optimization algorithm; (b) As an interval global optimization algorithm which consists in repeatedly executing the FIO algorithm. At every repetition the algorithm works on the set of boxes saved in the secondary list during the previous execution.

Table 3. Numerical results of the FIO algorithm with monotonicity test, $\epsilon = 10^{-6}$ and using bisection of the two widest dimensions of a box in one step as division rule

	Selection Criterion based on $p\overline{f^*}$					Selection Criterion based on $\underline{F}$				
Prob.	CPU	N_m	FX	GX	Fx	CPU	N_m	FX	GX	Fx
GP	0.07	2	173	173	43	11.75	5,867	57,185	30,554	14,296
S10	0.35	1	193	189	48	0.35	1	193	189	48
S7	0.25	1	193	188	48	0.25	1	193	188	48
S5	0.18	1	193	183	48	0.17	1	193	183	48
H6	2.59	31	1,965	1,715	491	2.34	20	1,677	1,558	419
H3	0.39	5	497	482	124	0.53	8	681	638	170
L3	0.27	1	101	101	25	0.33	2	129	125	32
L5	0.29	1	101	101	25	0.35	2	125	121	31
L8	0.09	1	153	130	38	0.09	1	153	130	38
C	0.08	3	193	190	48	0.35	63	1,245	815	311
C3	0.23	29	817	647	204	0.24	39	929	651	232
G2	0.02	1	113	32	28	0.01	1	113	32	28
G10	0.37	1	621	171	155	0.38	1	621	171	155
BR2	0.07	4	349	188	87	0.07	4	361	196	90
SRB2	0.01	1	89	54	22	0.01	1	89	54	22
RB2	0.01	1	89	57	22	0.01	1	89	57	22
RB10	0.33	1	441	244	110	0.33	1	441	244	110
P	0.01	1	101	33	25	0.01	1	101	33	25
TR	0.01	1	97	29	24	0.01	1	97	29	24
S3.1	0.10	3	413	294	103	0.05	1	153	143	38
MT	0.01	1	101	29	25	0.01	1	101	29	25
EX1	0.03	4	113	97	28	0.03	4	113	97	28
EX2	125.02	426	72,789	71,677	18,197	868.50	3,531	507,989	498,597	126,997
BR	0.05	2	189	161	47	0.05	2	189	161	47
KW	38.27	155	21,033	20,399	5,258	711.30	3,254	390,205	378,195	97,551
N2	31.41	740	53,821	28,123	13,455	31.33	740	53,821	28,123	13,455
N3-10	538.98	965	514,617	514,617	128,654	> 6h	$> 5 \cdot 10^4$	$> 2 \cdot 10^7$	$> 2 \cdot 10^7$	$> 6 \cdot 10^6$
R4	0.19	10	677	657	169	0.22	29	921	739	230
R8	4.52	5	2,237	2,127	559	4.52	5	2,241	2,131	560
HM3	0.90	109	1,333	876	333	0.09	1	101	100	25
HM4	2.61	96	2,113	1,712	528	2.12	64	1,657	1,447	414
BL	0.11	7	629	320	157	0.11	6	549	311	137
CHI	0.04	1	105	101	26	0.03	1	105	101	26
RT20	1.38	1	1,241	341	310	1.37	1	1,241	341	310

Algorithm 1 has been tested using two different selection criteria: based on $p\overline{f^*}$ and $\underline{F}$ parameters. The FIO algorithm with the parameter $p\overline{f^*}$ is more efficient because the value of N_m, which indicates the necessary number of boxes selected on each level of the search tree to reach the global solution, is smaller. The N_m parameter also determines the maximum width of the search tree. Therefore, using the $p\overline{f^*}$ parameter the memory complexity and the execution time is reduced, which helps to find a global solution quickly, even without using derivative information.

References

1. Berner, S. (1996), New Results in Verified Global Optimization. *Computing* 57(4), 323–343.
2. Casado, L.G. and García, I. (1999), Work Load Balance Approaches for Branch and Bound Algorithms on Distributed Systems. In: *Proceedings of the 7th Euromicro Workshop on Parallel and Distributed Processing*. University of Madeira, Funchal, Portugal, pp. 155–162.
3. Csendes, T. and Ratz, D. (1997), Subdivision Direction Selection in Interval Methods for Global Optimization. *SIAM Journal of Numerical Analysis* 34, 922–938.
4. Dixon, L. and Szego, G. (eds) (1975), *Towards Global Optimization*. Amsterdam: North-Holland Publishing Company.
5. Dixon, L. and Szego, G. (eds) (1978), *Towards Global Optimization 2*. Amsterdam: North-Holland Publishing Company.
6. Hammer, R., Hocks, M., Kulisch, U. and Ratz, D. (1995), *C++ Toolbox for Verified Computing I: Basic Numerical Problems: Theory, Algorithms, and Programs*. Springer-Verlag, Berlin.
7. Hansen, E. (1992), *Global Optimization Using Interval Analysis*, Vol. 165 of *Pure and Applied Mathematics*. Marcel Dekker, Inc., New York.
8. Henriksen, T. and Madsen, K. (1992), Use of a depth-first strategy in parallel Global Optimization. Technical Report 92-10, Institute for Numerical Analysis, Technical University of Denmark.
9. Horst, R. and Pardalos, P. (eds) (1995), *Handbook of Global Optimization*, Vol. 2 of *Noncovex optimization and its applications*. Dordrecht, Kluwer Academic Publishers, The Netherlands.
10. Horst, R. and Tuy, H. (1996), *Global Optimization. Deterministic Approaches* (third edition). Springer-Verlag, Berlin.
11. Ibaraki, T. (1988), Enumerative Approaches to Combinatorial Optimisation. *Annals of Operations Research* 11(1–4).
12. Ichida, K. and Fujii, Y. (1979), An Interval Arithmetic Method for Global Optimization. *Computing* 23: 85–97.
13. Knüppel, O. (1993a), BIAS-Basic Interval Arithmetic Subroutines. Technical Report 93.3, University of Hamburg.
14. Knüppel, O. (1993b), PROFIL-Programmer's Runtime Optimized Fast Interval Library. Technical Report 93.4, University of Hamburg.
15. Knüppel, O. and Simenec, T. (1993), PROFIL/BIAS Extensions. Technical Report 93.5, University of Hamburg.
16. Moore, R. and Ratschek, H. (1988), Inclusion Functions and Global Optimization II. *Mathematical Programming* 41: 341–356.
17. Neumaier, A. (1999), *Test functions*. http://solon.mat.univie.ac.at/∼vpk/math/funcs.html.
18. Ratschek, H. and Rokne, J. (1988), *New Computer Methods for Global Optimization*. Ellis Horwood Ltd., Chichester.
19. Ratz, D. (1992), Automatische Ergebnisverifikation bei globalen Optimierungsproblemen. Ph.D. thesis, University of Karlsruhe.

20. Ratz, D. and Csendes, T. (1995), On the Selection of Subdivision Directions in Interval Branch and Bound Methods for Global Optimization. *Journal of Global Optimization* 7: 183–207.
21. Shimano, Y., Harada, K. and Hirabayashi, R. (1997), Control schemes in a generalized utility for parallel Branch-and-Bound algorithms. In: *11th International Parallel Symposium (IPPS'97)*. pp. 621–627.
22. Törn, A. and Žilinskas, A. (1989), *Global Optimization*, Vol. 3350. Springer-Verlag, Berlin.
23. Walster, G., Hansen, E. and Sengupta, S. (1985), Test results for global optimization algorithm. *SIAM Numerical Optimization 1984*, pp. 272–287.