BIT
200?, Vol. ??, No. ??, pp. ???–???

0006-3835/98/3804-0101 $12.00

A Heuristic Rejection Criterion in Interval Global Optimization Algorithms*

L.G. CASADO[1], I. GARCÍA[2] and T. CSENDES[3]

[1,2] *Department of Computer Architecture and Electronics, Almería University, 04120, Almería, Spain. emails: [1] leo@ace.ual.es, [2] inma@ace.ual.es*

[3] *Institute of Informatics, University of Szeged, H-6701 Szeged, Pf. 652, Hungary. email: csendes@inf.u-szeged.hu*

Abstract.

This paper investigates the properties of the inclusion functions on subintervals while a Branch-and-Bound algorithm is solving global optimization problems. It has been found that the relative place of the global minimum value within the inclusion interval of the inclusion function of the objective function at the actual interval mostly indicates 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 a numerical study completes the investigation.

AMS subject classification: 65K, 90C.
Key words: Global optimization, Branch-and-Bound Algorithm, Inclusion Function.

1 Introduction

This paper investigates variants of a Branch-and-Bound (B&B) algorithm [5, 8] solving the bound constrained global optimization problem [4]:

$$\min_{x\in X} f(x), \tag{1.1}$$

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 is denoted by f^*, a global minimizer point by x^*, and the set of global minimizer points on X by X^*.

Real bounded and closed interval vectors are denoted by $X = [\underline{X}, \overline{X}]$. The set of compact intervals is denoted by $\mathbb{I} := \{[a, b] \mid a \leq b,\ a, b \in \mathbb{R}\}$. The width of the interval X is defined by $w(X) = \max_{i=1..n} w(X_i)$. The midpoint is $m(X) = (m(X_1), m(X_2), \ldots, m(X_n))^T$. Our implementation is based on the Moore-Skelboe interval B&B algorithm [8].

A function $F : \mathbb{I}^n \to \mathbb{I}$ is called an *inclusion function* of f in $X \subseteq \mathbb{R}^n$, if $x \in X$ implies $f(x) \in F(X)$. F is said to be an *inclusion isotone function* over X if $\forall Y, Z \in \mathbb{I}^n(X)$ $Y \subseteq Z$ implies $F(Y) \subseteq F(Z)$. F is an order α inclusion function

*This work was supported by the Ministry of Education of Spain (CICYT TIC99-0361), and by the Grants FKFP 0739/97, 0449/99, OTKA T 016413, T 017241 and T 32118.

Algorithm 1.1 : Basic Interval B&B Global Optimization Algorithm.

<u>Funct</u> GlobalOptimize(X, F, ϵ, T, Q)

$T{:=}\{X\}$; $Q{:=}\{\}$ — *Work and Final trees*
$\tilde{f}{:=}\overline{F}(X)$; — *Upper bound for f^**
<u>while</u> $(T \neq \{\emptyset\})$
  $B{:=}$ SelectNextBox(T); — $\underline{F}(B) = \min\{\underline{F}(B^i),\ \forall B^i \in T\}$
  $T{:=}T - B$;
  $\tilde{f}{:=}\min\{\tilde{f}, \overline{F}(m(B))\}$; — *Trying to improve the upper bound of f^**
  $T{:=}$CutOffTest$(T, \tilde{f})$; — $\forall X^i \in T, \tilde{f} < \underline{F}(X^i) \Rightarrow T := T - \{X^i\}$
  $Q{:=}$CutOffTest$(Q, \tilde{f})$;
  Subdivide $(B, B^1, ..., B^s)$; — *In this work $s = 2^n$*
  <u>for</u> i:=1 **<u>to</u>** s
    InsertBoxTree $(B^i, T, Q, \epsilon, \tilde{f})$
<u>return</u> Q

Algorithm 1.2 : Subroutine of Algorithm 1.1.

<u>Proc</u> InsertBoxTree$(B, T, Q, \epsilon, \tilde{f})$

<u>if</u> $(\tilde{f} < \underline{F}(B))$ **<u>return</u>** ; — *Traditional rejection criterion*
<u>if</u> ($w(B) < \epsilon$) $Q{:=}Q + B$; — *Store box in the final tree*
<u>else</u> $T{:=}T + B$; — *Store box in the work tree*

(*α-convergent*) of f over X if $\forall Y \in \mathbb{I}^n(X) \quad w(F(Y)) - w(f(Y)) \leq Cw(Y)^\alpha$, where C and α are positive constants.

In the present work bisection in all dimensions is applied. Boxes are sorted in a nondecreasing 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. A balanced binary tree is applied for saving the set of generated subproblems. The termination rule is $w(Y) < \epsilon$ for all the remaining boxes. We use the cut-off and the middle point test. The basic algorithm is Algorithm 1.1-1.2. The *SelectNextBox* procedure chooses the first box B of the tree T, for which $\underline{F}(B) = \min_{B^i \in T} \underline{F}(B^i)$.

Suppose that for a given function $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$, 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 *RejectIndex* as:

$$pf^*(Y) = \frac{f^* - \underline{F}(Y)}{\overline{F}(Y) - \underline{F}(Y)} \in [0, 1],\ Y \subseteq X.$$

This parameter was designed mainly based on the following: Traditionally an interval X with the minimal value of $\underline{F}(X)$ was considered as the best candidate to contain a global minimum. However, usually the larger the interval X, the

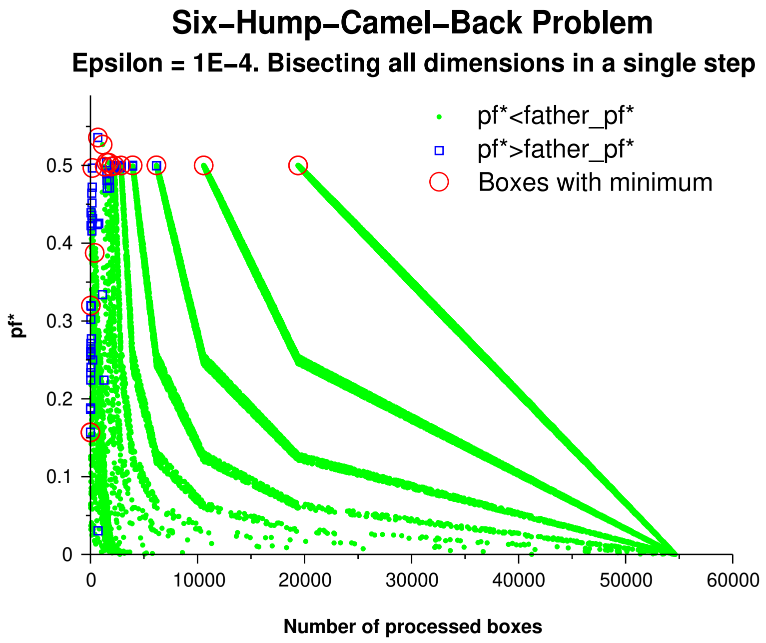

Figure 1.1: Values of pf^* while the Algorithm 1.1-1.2 is running on the two dimensional Six-Hump-Camel-Back problem. The “curves” usually contain boxes generated from boxes located in the previous “curve”. At a given iteration number, the figure shows that in that phase of the algorithm boxes with different RejectIndex value exist. Notice that no box contains a minimizer point after 20,000 boxes are processed.

larger the overestimation of $f(X)$ obtained in $F(X)$. Therefore a box could be considered as a good candidate to contain a global minimum just because it is larger than the others. In order to compare subintervals with different size we normalize the distance between f^* and $\underline{F}(X)$. The idea behind pf^* is simply that we expect the overestimation to be symmetric, i.e. the overestimation above $f(X)$ is closely equal to the overestimation below $f(X)$, for small subintervals 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. Figure 1.1 shows the values of pf^* obtained while solving the two dimensional Six-Hump-Camel-Back problem with two global minimizer points at (0.0899, -0.7126) and (-0.0899, 0.7126). For this problem it is clear that as the branch-and-bound algorithm evolves only those intervals containing a minimizer point have the greater pf^* values.

Figure 1.2 shows the results of evaluating pf^* for 64×64 equal size intervals

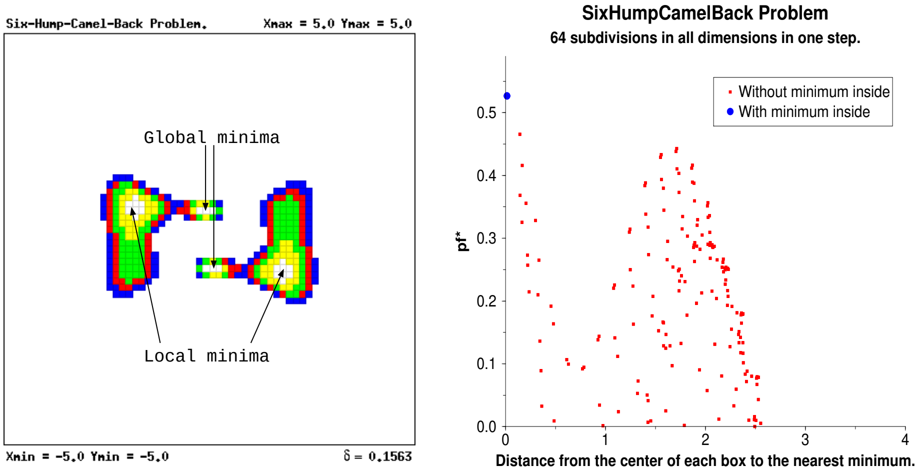

Figure 1.2: In the left hand figure, the intervals that have been eliminated correspond to blank positions. The brighter the boxes, the greater the pf^* values. In the right hand diagram, the values of pf^* are shown as a function of the distance to the nearest global minimizer point.

generated by dividing the whole search space of the Six-Hump-Camel-Back problem. This figure indicates that the higher RejectIndex values belong to intervals close to a minimizer point.

The *RejectIndex* 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 [1], it has been shown that using *RejectIndex*, an almost perfect work load balance for parallel implementations of the Interval B&B algorithms can be obtained.

It is well known that the majority of the computational work in interval B&B algorithms is made on boxes belonging to the close neighbourhood of minimizer points. In this work, it will be shown that *RejectIndex* can be used in a heuristic rejection criterion that improves the efficiency of interval B&B algorithms. The motivation of the development of such a storage saving heuristic was that, in many cases, hard problems cannot be solved by interval techniques, since the computer storage to save the emerging subproblems is not large enough. Madsen and Zertchaninov reported a stochastic, thus heuristic version of the interval optimization algorithm that aimed also to solve high memory complexity problems [7]. According to our experiments large RejectIndex value for the interval X indicate that it is usually close to a local or global minimizer point [2]. Now we investigate whether and to what extent this surprisingly easy to obtain information about the location of the minima is reliable, and how efficiently it can be applied to improve the efficiency of B&B algorithms by deleting those subintervals which cannot contain a minimizer according to the pf^* values.

Algorithm 2.1 : The new rejection criterion.

Proc InsertBoxTree($B, B', B'', T, Q, \epsilon, \tilde{f}, DecTh$)
var static SecondStage := False;

1. **if** ($\tilde{f} < \underline{F}(B)$) **return** ; *Traditional rejection criterion*
2. **if** (SecondStage = False **and** $pf^*(B) > pf^*(B')$) SecondStage = True;
3. **if** (SecondStage = True **and** $pf^*(B) < pf^*(B')$
   and $pf^*(B') < pf^*(B'')$
   and $pf^*(B'') - pf^*(B) > DecTh$) **return** ; *New rejection criterion*
4. **if** ($w(B) < \epsilon$) $Q{:=}Q + B$; *Store box in the final tree*
5. **else** $T{:=}T + B$; *Store box in the work tree*

2 Improving a Branch-and-Bound algorithm

During the execution of the B&B algorithm, two stages can be distinguished. The initial stage, if it exists, finishes when an increase of the pf^* value appears. Based on the experiments, we can state that in the second stage if two consecutive offsprings B and B' of a box B'' exhibit a significant decrease of the pf^* value then this box does not contain a global optimizer point, and therefore it can be rejected.

Increasing *RejectIndex* 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 *RejectIndex* values rarely were connected to interval sequences converging to points with function values above f^*.

Algorithm 1.1-2.1 uses these findings for an efficient rejection. The larger the $DecTh$ value the smaller the number of rejected boxes. With small $DecTh$ values we could reject boxes containing minimizer points. $DecTh = 1$ inhibits the rejection ideas and with $DecTh = 0$ only sequences of increasing pf^* values are allowed.

The suggested rejection criterion is based on the empirical experience that the overestimation of $F(X)$ remains closely proportional to $w(f(X))$ both above and below $f(X)$. In this case the patterns to be recognized by the rejection criterion are related to boxes not containing global minimizer points, since then the pf^* values must tend to decrease.

By the following example we show that there exists isotone and α-convergent inclusion functions such that Algorithm 1.1-2.1 does not converge to x^*.

EXAMPLE 2.1. To illustrate a case when the new rejection rule gives a false decision, consider the function $f(x) = x(x-2)$. Its minimum is -1, which is attained only at $x^* = 1$. Four intervals are studied which can be actual intervals of our interval subdivision procedure, starting from [0, 16]. The ranges of $f(x)$ on these intervals are summarized in Table 2.1. In the next two columns

Table 2.1: Intervals (X_i), ranges ($f(X_i)$), natural interval extension inclusions ($F(X_i)$), and the constructed inclusion function values ($G(X_i)$) for Example 2.1.

i	X_i	$f(X_i)$	$F(X_i)$	pf_F^*	$G(X_i)$	pf_G^*
0	[0, 16]	[-1, 224]	[-32, 224]	0.1211	[-200, 224]	0.4693
1	[0, 8]	[-1, 48]	[-16, 48]	0.2343	[-200, 48]	0.8024
2	[0, 4]	[-1, 8]	[-8, 8]	0.4375	[-32, 8]	0.7750
3	[0, 2]	[-1, 0]	[-4, 0]	0.7500	[-4, 0]	0.7500

we give the inclusion function values obtained by natural interval extension, and the respective pf^* figures. We can see that for this often used inclusion function there is a starting stage, followed by an increase in the pf^* values — corresponding to our experiences. These pf_F^* values do not allow a rejection. In the next column a new inclusion function G is given, that differs from F only in the lower bounds for three intervals, i.e. it has all the properties of F: inclusion isotonicity and α-convergence. The respective pf_G^* values are given in the last column.

By the later construction and using a $DecTh \leq 0.05$, the new rejection routine will set the *SecondStage* parameter true when X_1 is processed, and the interval X_3 will be fathomed, since the pf_G^* values are decreasing for the last three subintervals, and also the difference in the pf_G^* parameter values is larger than the set threshold value, regarding the intervals X_1 and X_3. Hence, the interval X_3 will not be processed further, and in this way Algorithm 1.1-2.1 cannot converge to the point $x^* \in X_3$. Since for all the four selected intervals $\underline{G}(X_i) \leq \underline{F}(X_i)$ holds, the respective intervals will be selected not later than the same algorithm would do for the inclusion function $F(X)$. A similar construct can be used for problems of larger dimension and with larger finite number of global minimizer points.

3 Numerical results

The numerical tests were carried out on a Pentium-II PC (233 MHz., 256 MBytes RAM) running the Linux operating system. The programs were coded in C. The inclusion functions were implemented via the BIAS routines [6]. The standard time unit (the CPU time required to evaluate the Shekel 10 test function 1000 times at $(4.0, 4.0, 4.0, 4.0)^T$) was 0.009 seconds. We have used the following seven standard global optimization test problems: Six-Hump-Camel-Back (SHCB), Goldstein-Price (GP), Hartman-3 (H3), Levy-3 (L3), Shekel-5 (S5), Shekel-7 (S7) and Shekel-10 (S10). The search regions were the same as in other numerical tests [8]. In our tables, the following notation is used for column headers:

ϵ The stopping criterion parameter ($w(Y) < \epsilon$).
CPU Execution time in seconds.

Table 3.1: Numerical results of the Algorithm 1.1-1.2 with one subdivision in all dimensions.

Problem	ϵ	CPU	EX	Ex	FT	MT	NBM
SHCB	10^{-2}	0.15	8,597	2,149	424	856	19
	10^{-3}	0.68	33,413	8,353	6,614	4,322	27
	10^{-4}	4.77	218,469	54,617	52,866	34,986	33
GP	10^{-2}	0.81	32,145	8,036	12,100	3,069	31
	10^{-3}	3.67	169,201	42,300	9,512	14,535	43
	10^{-4}	16.96	689,085	172,271	135,096	90,773	59
H3	10^{-2}	0.52	10,233	1,279	2,505	924	7
	10^{-3}	14.99	283,409	35,426	71,892	28,048	10
	5×10^{-4}	45.95	858,545	107,318	202,894	85,812	11
L3	10^{-2}	0.57	3,629	907	501	291	86
	10^{-3}	5.31	32,749	8,187	7,680	5,005	122
	10^{-4}	40.42	246,553	61,638	60,615	40,301	149
S5	10^{-2}	0.01	161	10	1	1	10
	10^{-3}	0.03	705	44	43	27	14
	10^{-4}	0.42	9,009	563	1,021	400	17
S7	10^{-2}	0.03	385	24	9	15	10
	10^{-3}	0.27	4,609	288	480	200	14
	10^{-4}	7.50	123,905	7,744	20,810	6,241	17
S10	10^{-2}	0.04	417	26	18	16	10
	10^{-3}	0.46	5,985	374	601	252	14
	10^{-4}	12.15	157,105	9,819	26,038	7,986	17

EX	Number of $F(X)$ evaluations.
Ex	Number of $F([x,x])$ evaluations (equal to the number of iterations).
FT	Number of members in the Final Tree.
MT	Maximum number of nodes necessary in the working Tree.
NBM	Number of Boxes including any Minimizer points, during execution.
$\overline{f^*}$, $\underline{f^*}$	Upper and lower bounds of f^*, $\underline{f^*} = \min\{\underline{F}(B^i),\ \forall B^i \in Q\}$.

In Table 3.1, results for the basic algorithm are shown. The most important property is that the number of boxes containing minimizer points (NBM), is very small compared to the number of evaluated boxes (EX) and to the number of iterations (Ex). These differences grow further with the accuracy. For the Hartman-3 problem, the program was executed only with $\epsilon = 5\times10^{-4}$, because it run out of memory with $\epsilon = 10^{-4}$. In Table 3.2, the results with this new heuristic are shown. Comparing Tables 3.1 and 3.2, it can be seen that, in general, all the parameter values are improved. The efficiency of the proposed new algorithm is some 80% better than the basic algorithm on hard problems, and thus for fixed computer resources more accurate solutions can be obtained.

Table 3.2: Numerical results of the Algorithm 1.1-2.1 using the known f^* value and $DecTh = 0.04$.

Problem	ϵ	CPU	EX	Ex	FT	MT	NBM
SHCB	10^{-2}	0.02	1,101	275	22	78	19
	10^{-3}	0.07	3,533	883	712	406	27
	10^{-4}	0.48	23,245	5,811	5,614	3,412	33
GP	10^{-2}	0.08	3,921	980	83	250	31
	10^{-3}	0.14	6,781	1,695	909	451	43
	10^{-4}	1.42	60,321	15,080	14,265	8,441	59
H3	10^{-2}	0.18	3,345	418	903	271	7
	10^{-3}	1.67	33,025	4,128	3,321	2,475	10
	5×10^{-4}	3.06	59,593	7,449	8,289	4,324	11
L3	10^{-2}	0.47	3,097	774	235	251	86
	10^{-3}	1.30	8,389	2,097	890	680	122
	10^{-4}	4.93	31,277	7,819	5,866	3,882	149
S5	10^{-2}	0.01	161	10	1	1	10
	10^{-3}	0.03	705	44	18	27	14
	10^{-4}	0.29	6,513	407	482	309	17
S7	10^{-2}	0.03	385	24	5	15	10
	10^{-3}	0.24	4,081	255	318	183	14
	10^{-4}	1.91	32,545	2,034	1,203	1,392	17
S10	10^{-2}	0.04	417	26	7	16	10
	10^{-3}	0.38	4,913	307	357	216	14
	10^{-4}	2.57	33,313	2,082	1,379	1,375	17

Table 3.3: The average (abbreviated as Ave.1) of the data in Table 3.1 are summarized together with the relative values (abbreviated as Rel.Ave.2 = Ave.2 / Ave.1) compared to the average values of the data in Table 3.2.

ϵ	CPU		EX		Ex	
	Ave.1	Rel.Ave.2	Ave.1	Rel.Ave.2	Ave.1	Rel.Ave.2
10^{-2}	0.30	38.97%	7,938	22.36%	1,776	20.17%
10^{-3}	3.63	15.07%	75,724	11.59%	13,567	9.91%
10^{-4}	18.31	11.44%	328,953	10.72%	59,139	9.83%

In Table 3.3, the average of the CPU time and the number of EX and Ex evaluations are shown for all problems in Table 3.1 as Ave.1. For the data in Table 3.2 the relative values are compared to the ones in Table 3.1 (Rel.Ave.2 = Ave.2/Ave.1). The improvement grows with the accuracy till over 88% (for $\epsilon = 10^{-4}$). The global minima were located in all the tested cases.

This heuristic rejection criterion has been tested also on the difficult EX2 problem [3]. This time the bisections were made along the widest side, the ter-

Table 3.4: Numerical results for EX2 problem without and with the new rejection criterion ($DecTh = 0.004$).

ϵ	CPU	EX	Ex	FT	MT	NBM	$[\underline{f^*}, \overline{f^*}]$
0.50	53.39	377,767	188,883	55,029	29,495	25	[0.085849, 0.214625]
0.10	-	-	-	-	-	-	-
0.50	11.57	83,153	41,576	6,374	5,915	25	[0.085849, 0.214625]
0.10	37.66	267,701	133,850	10,193	23,799	35	[0.171298, 0.212732]
0.05	58.53	415,505	207,752	9,548	33,013	40	[0.190558, 0.212581]
0.01	178.75	1,250,593	625,296	133,600	70,992	63	[0.207572, 0.212465]

mination criterion was $w(F(X)) \leq \epsilon$, and the value of f^* was not used. Table 3.4 contains the numerical results for both algorithms. The problem with $\epsilon = 0.1$ could not be solved with the old technique due to extensive memory complexity. For the new algorithm the results for $\epsilon = 0.5$ indicate closely 80% improvement in all the major efficiency indicators. It is even more important that the improved algorithm is able to solve the problem with acceptable precision, while it was impossible for the basic method. Although our new procedure is simple, these results are comparable or better than those in [3]. These numerical results confirm that the earlier findings on the standard test problems are valid for more hard problems. The achieved nice efficiency improvements of decreasing the representative efficiency indicators, by closely one order of magnitude, are the more valuable, since these were obtained by saving the reliability: all the subintervals containing minimizer points were found also by the improved technique.

4 Conclusions and future work

We have found that the minimizer points are located in boxes with large *RejectIndex* values. The subintervals generated from them will contain a minimizer point only if the *RejectIndex* does not decrease more than a threshold value for three consecutive successor intervals. Computational tests on a set of functions have shown that the introduced heuristic rejection criterion substantially improves the efficiency of the underlying Branch-and-Bound algorithm and these improvements were larger when more accurate results were asked for. Numerical results indicate that the new rejection test may make it possible to solve hard problems, which are otherwise insolvable with available computational resources. For difficult to solve problems the new methodology can decrease the computational effort by as much as one order of magnitude.

The algorithm requires the knowledge of the global minimum, but the latter can also be estimated by the actual 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 subintervals into a file, and their later offline processing. The new algorithm is also suitable for non-differentiable problems where advanced accelerating devices can not be applied.

Acknowledgments.

The authors would like to thank to the referees and Kay Madsen (editor) for their useful remarks and suggestions.

REFERENCES

1. L. G. Casado and I. García, *New load balancing criterion for parallel interval global optimization algorithms*, in Proceedings of the 16th IASTED International Conference on Applied Informatics, Garmisch-Partenkirchen, Germany, February 1998, pp. 321–323.
2. L. G. Casado, I. García, and T. Csendes, *A new multisection technique in interval methods for global optimization*, Computing, 65 (2000), pp. 263–269.
3. T. Csendes and D. Ratz, *Subdivision direction selection in interval methods for global optimization*, SIAM Journal of Numerical Analysis, 34 (1997), pp. 922–938.
4. R. Horst and P. Pardalos, eds., *Handbook of Global Optimization*, vol. 2 of Noncovex optimization and its applications, Kluwer Academic Publishers, 1995.
5. R. B. Kearfott, *Rigorous Global Search: Continuous Problems*, Kluwer Academic Publishers, 1996.
6. O. Knüppel, *BIAS-Basic Interval Arithmetic Subroutines*, Tech. Rep. 93.3, University of Hamburg, 1993.
7. K. Madsen and S. Zertchanninov, *A new branch and bound method for global optimization*, Tech. Rep. IMM 05/98, Technical University of Demmark, Copenhagen, 1998.
8. H. Ratschek and J. Rokne, *New computer methods for global optimization*, Ellis Horwood Lmt.(John Willey & Sons), 1988.