Journal of Global Optimization **16:** 371–392, 2000.
© 2000 *Kluwer Academic Publishers. Printed in the Netherlands.*

Multisection in Interval Branch-and-Bound Methods for Global Optimization I. Theoretical Results*

ANDRÁS ERIK CSALLNER[1], TIBOR CSENDES[2] and MIHÁLY CSABA MARKÓT[3]

[1]*Department of Computer Science, Juhász Gyula Teachers Training College, Boldogasszony sgt. 4, Szeged, Hungary (e-mail: csallner@jgytf.u-szeged.hu);* [2]*Department of Applied Informatics, József Attila University, Árpád tér 2, Szeged, Hungary (e-mail: csendes@inf.u-szeged.hu);* [3]*Institute of Informatics, József Attila University, Árpád tér 2, Szeged, Hungary (e-mail: markot@inf.u-szeged.hu)*

(Received 27 April 1999; accepted in revised form 17 November 1999)

Abstract. We have investigated variants of interval branch-and-bound algorithms for global optimization where the bisection step was substituted by the subdivision of the current, actual interval into many subintervals in a single iteration step. The convergence properties of the multisplitting methods, an important class of multisection procedures are investigated in detail. We also studied theoretically the convergence improvements caused by multisection on algorithms which involve the accelerating tests (like e.g. the monotonicity test). The results are published in two papers, the second one contains the numerical test result.

Key words: Branch-and-bound method; Global optimization; Interval arithmetic; Multisection; Accelerating devices

1. Introduction

The aim of this paper is to analyze algorithms solving the unconstrained global optimization problem. In general, we will assume that a nonempty bounded closed n-dimensional interval or box $X \subset \mathbb{R}^n$ containing all global minimizers x^* of the (in most cases continuous) objective function $f : \mathbb{R}^n \to \mathbb{R}$ can always be given. Considering real-life problems this means practically no restrictions on the type of problems considered. Keeping this argumentation in view, the bound constrained global optimization problem has the following form:

$$\min_{x \in X} f(x) . \tag{1}$$

The algorithms considered are based on interval arithmetic [1, 10, 14, 17]. We shall

* The work has been supported by the Grants AMFK 398/95, FKFP 0739/97, OTKA F 025743, T 016413, T 017241, and MKM 75/96.

denote the inclusion function of the objective function f by $F : \mathbb{I}^n \to \mathbb{I}$, i.e., for $\forall Y \in \mathbb{I}^n$ and $\forall y \in Y\ f(y) \in F(y)$, where $\mathbb{I}$ stands for the set of all bounded closed real intervals. In other words, $f(Y) \subseteq F(Y)$ where $f(Y)$ is the range of f over Y. The lower and upper bounds of an interval $Y \in \mathbb{I}^n$ are denoted by $\mathrm{lb}Y$ and $\mathrm{ub}Y$, respectively, and the width by $w(Y)$: $w(Y) - \max_i(\mathrm{ub}Y_i - \mathrm{lb}Y_i)$. $\mathbb{I}(X)$ stands for all $Y \in \mathbb{I}^n$ such that $Y \subseteq X$. Two important properties of inclusion functions of continuous functions are:

DEFINITION 1. F is said to be an isotone inclusion function over X if for $\forall Y$, $Z \in \mathbb{I}(X)$ $Y \subseteq Z$ implies $F(Y) \subseteq F(Z)$.

DEFINITION 2. We call F α-convergent over X if for $\forall Y \in \mathbb{I}(X)$ $w(F(Y)) - w(f(Y)) \leqslant cw^{\alpha}(Y)$, where c is a positive constant.

Much effort has been made to improve the convergence speed of interval methods for global optimization in the last few decades to enable these reliable methods to solve real-life problems [4–6, 7, 9, 13, 18–20]. The main part of this paper develops further an idea [2, 3, 10, 14], subdividing the current subproblem into many ($s > 2$) smaller problems in a single step in contrast to traditional bisection, where two new subintervals are always produced.

1.1. INTERVAL BRANCH-AND-BOUND METHODS

Branch-and-bound algorithms are based on successive subdivision of the set of feasible solutions. They use various branching rules mostly based on bounds on the objective function values to select a promising subset that might contain a global solution. Other bounds, e.g. those on the gradient of $f(x)$, are used to exclude subsets that surely does not contain any solutions. Thus, the basic branch-and-bound principle usually requires both lower and upper bounds on the function values over a set of the search domain which can be an interval, as well. Interval arithmetic provides these bounds.

The general algorithm can be formulated as follows:

Model Algorithm

Step 1. Let L be an empty list, set the current box $A := X$, and the iteration counter $k := 1$.

Step 2. Subdivide A into a finite number of subsets A_i satisfying $A = \cup A_i$ so that $\mathrm{int}(A_i) \cap \mathrm{int}(A_j) = \emptyset$ for all $i \neq j$ where 'int' denotes the interior of a set.

Step 3. Add the subintervals $\{A_i\}$ to K.

Step 4. Discard certain elements from L that cannot contain a global minimizer.

Step 5. Choose a new $A \in L$ and delete if from the list, $L := L \backslash \{A\}$.

Step 6. While termination criteria do not hold set $k := k + 1$ and go to Step 2.
Step 7. Stop.

The algorithm's iterative part begins with Step 2, the subdivision of a given current interval A. This step is essential when trying to accelerate the model algorithm and is investigated in subsection 1.3 and section 2 in details.

The outlined algorithm manages a list L which contains intervals whose union includes all global minimizers of the considered problem. One of the generally used tools is to update an upper bound on the global minimum f^*, and deleting the intervals having a larger objective function lower bound than this stored value (cut-off test).

It is convenient to manage another list, L_{out} to collect all intervals from the list L where the bounds are tight or where the boxes are narrow enough. Thus, we can terminate the algorithm, e.g., when our list L becomes empty.

1.2. ACCELERATING INTERVAL SUBDIVISION METHODS

Many tests exist to accelerate interval subdivision methods in general. Due to possible overestimations, they do not influence the worst case behavior and hence the theoretical convergence speed of a particular algorithm. These tests can lead to better results on wide classes of optimization problems, but in general, neither can these classes be determined explicitly, nor can the worst case speed be improved. Some of these tests, usually called accelerating devices [17], are listed below not intending to be exhaustive.

The most widely used accelerating device is the *cut-off test*: find as small objective function value upper bounds as possible. Based on such a bound and the lower bounds computed for the elements of L, many subintervals can usually be discarded from the list. However, for an objective function that is 'flat' around the global minimizer points, this device does not help much. A traditional local search procedure can supply good upper bounds for the cut-off test.

The other widely used accelerating tool is the *monotonicity test*. Of course, subsets where the objective function is strictly monotonous cannot contain stationary points inside. This device is not efficient for an objective function that has several saddle points and local minimizers. The same function can cause problems for the *concavity test* that discards intervals over which the objective function is strictly concave in a variable (since these intervals cannot contain a minimizer point inside). The subdivision selection rules are described in [16].

Further ways to increase efficiency are the use of different inclusion functions, e.g., the slope functions [11, 14, 15] or the centered forms with Baumann centers [2]. This is a very interesting way of improving inclusions and further investigations will most probably result in nicer worst case results for certain problem classes. This paper, however, does not deal with these tools because we want to concentrate on the effects of multisection.

Some algorithmic modifications can change the behavior of a method. These are the modifications to Step 2 and Step 5 of the model algorithm which correspond to the branching in branch-and-bound algorithms. In Step 2 we can vary the direction and the number of the cuts. We could certainly also change the proportions of the arising subintervals but the information supplied by interval arithmetic is usually not enough itself to make proper decisions. In Step 5 the way to choose a new current box has to be determined.

1.3. THE INTERVAL SELECTION RULE

In each iteration cycle a new interval, the current box has to be picked up from the list L for subdivision. Some information on the intervals can be stored in L or can be expressed implicitly by the ordering of the list elements. These informations have to be used to make the decision concerning which interval to choose.

The two most widely used variants are to set the current box be the list element with the smallest inclusion function lower bound [17], and the other is to choose the 'oldest' interval from the list which corresponds to one of the widest ones [10]. These rules change also the theoretic convergence properties of the model algorithm [17]. To be able to talk about these modifications we shall denote the algorithm using the former rule the Moore-Skelboe algorithm and that applying the latter one the Hansen algorithm. For both methods it is assumed that the current box is bisected in Step 2 through one of the longest edges and never discard any element in Step 4.

For the two considered algorithms the following assertion holds [17].

THEOREM 1. *If* $w(Y_i) \to 0$ *implies* $w(F(Y_i)) \to 0$ *or* $w(Y) \to 0$ *implies* $w(F(Y)) - w(f(Y)) \to 0$ *then both the Moore-Skelboe and the Hansen algorithms converge to the global minimum:* $\min_{Y \in L \cup A} \mathrm{lb}F(Y) \to f(x^*)$, *where* x^* *denotes one of the global minimizers.*

The main question is the convergence speed of these algorithms. Although the Moore-Skelboe method has turned out to be faster in practice, it is slower than the Hansen algorithm regarding the worst case behavior [7, 17]. The latter fact is reflected in the following two theorems [17]:

THEOREM 2. *The Moore-Skelboe algorithm converges arbitrary slowly if the applied inclusion function is not inclusion isotone.*

This restriction is not valid for the Hansen method as we have proved in [7]. Moreover, a theoretical worst case upper bound can be given for the difference between the global minimum and the computed lower bound after k iteration cycles [7].

THEOREM 3. *If F is the inclusion function of the objective function f applied in the Hansen algorithm and F is of order α then*

$$\mathrm{lb}f(X) - \mathrm{lb}F(A) \leqslant c(2w(X))^{\alpha}(k+1)^{-\alpha/n}, \tag{2}$$

where c is the positive constant of the α-convergence and A the current box of the kth cycle in the algorithm.

The same result was proved for the Moore-Skelboe algorithm when assuming inclusion isotonicity for the inclusion function [7].

THEOREM 4. *If F is an isotone inclusion function of the objective function f applied to the Moore-Skelboe algorithm and F is of order α then (2) holds where c is the positive constant of the α-convergence and A the current box of the kth iteration cycle in the algorithms.*

(2) states an exponential convergence speed with respect to the number of variables and no results have been published up to now proving a better upper bound on interval methods for global optimization. Although some noninterval methods have better practical convergence speed properties, they do not exploit global information and thus do not assure convergence for such a wide class of problems. In contrast to that, interval arithmetic provides lower and upper bounds on the function value over a whole set of points, represented by boxes, providing global information.

2. Multisection and Multisplitting

As it can be seen in the joint paper [16] too, it can be important to utilize any information on the objective function to decide which direction for the bisection should be chosen. If we revert to our first assumption, namely, the objective function can be any $f : \mathbb{R}^n \to \mathbb{R}$ real-valued function for which an evaluation routine is available, then we have to find other ways to accelerate our algorithms. One way to do so not discussed theoretically yet: changing the number of subintervals generated by a subdivision or in other words using multisection in a single step.

The idea of multisection, i.e., multiple bisection, arose in [2, 10, 19] where more than one bisection was made at a single iteration cycle. For serial algorithms the triple bisection was experimentally found to be the most efficient, while for parallel methods it was the double bisection. Thus, the current box is multisected into $2^3 = 8$ or $2^2 = 4$ subboxes (see Figure 1 for the 3-dimensional case) determining all of the bisection directions prior to the first cut at that iteration cycle.

The results are convincing, however, additional calculations have to be made for the choices of all bisection directions, and that consumes time. On the other hand, if the directions are determined earlier, less information is available, and subboxes

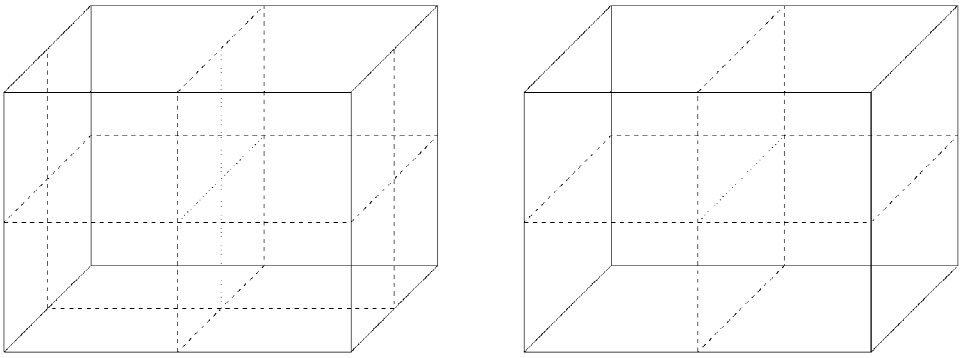

Figure 1. Two types of multisection: a triple bisection into $s = 2^3 = 8$ subintervals in a single iteration cycle, and as double bisection providing $s = 2^2 = 4$ subintervals.

which could have been discarded from further investigations are processed. We can save computations if we use only the most promising direction for subdivision. This type of multisection, when s equal size subintervals are produced will be called *multisplitting* in the sequel (Figure 2).

Both ideas above have something in common: They utilize the information on many smaller subboxes, and they are not investigating the larger boxes in the search tree. Let us consider this on an example on multisection.

Let us assume that we simply bisect the current box A into two subboxes A_1 and A_2, then the resulting A_1 into A_{11} and A_{12} and A_2 into A_{21} and A_{22}, respectively. If we assume that the particular bisection directions are the same for bisection and multisection, after a single 2-multisection step we get the four subboxes A_{11}, A_{12}, A_{21}, and A_{22}. Obviously, the function evaluations over these subboxes provide more information than the function evaluations over their predecessors, hence, making the function evaluations at A_1 and A_2 unnecessary, multisection can accelerate the

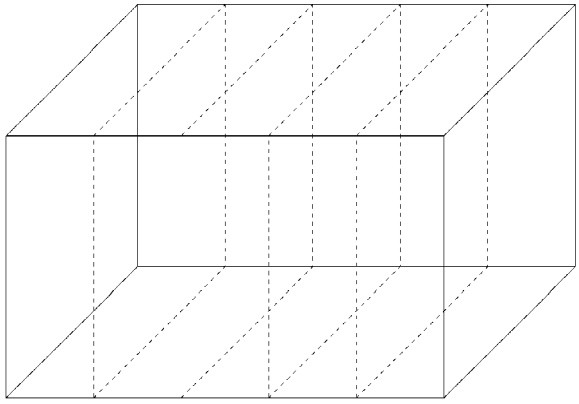

Figure 2. Multisplitting is a special multisection where s equal size subintervals are produced (here $s = 5$, i.e. the 5-splitting is shown).

method. On the other hand, however, we could have already discarded A_1 and A_2, saving the calculations for the smaller subboxes. If only one of them could have been discarded, then we have the same number of subboxes to be investigated as in the bisecting case. The decisive question is, whether there occur more advantageous than disadvantageous cases. The principle is the same for a multisplitting algorithm, apart from some unnecessary computations for the subdivision directions using multisection.

The process of subdivisions can be considered as a tree, as well, denoting the root with the initial box X, and the successors of a box with the subboxes it breaks up to. For a bisection method this is a binary tree and omitting certain levels from that tree we get the search tree of a multisection algorithm.

2.1. THE MULTISPLITTING ALGORITHM

To investigate the convergence speed of the multisplitting algorithm we restate some results from [7] extending them for the multisplitting case.

In the following we will consider the *multisplitting algorithm* which differs from the Hansen method only in splitting the current box A in Step 2 through the longest edge into s equal size subintervals where s can be any natural number greater than one. Without loss of generality we assume in the sequel that X is an *s-quasi hypercube*, i.e., every edge is shorter than s times the length of any other edge (if any). Expressed by formulas, if $X = X_1 \times \cdots \times X_n$ where the X_i $(i = 1, \ldots, n)$ interval denotes the ith projection of X, then $w(X_i) < sw(X_j)$ $(\forall i,\ j = 1, \ldots, n)$. Note that this assumption does not influence our theoretical investigations, since after some problem-specific constant number of iteration cycles X breaks down to s-quasi hypercubes which can be treated separately. Its importance, however, is reflected in the following notion of levels of iteration cycles for the multisplitting algorithm we define below.

DEFINITION 3. We say that iteration cycle number k is on the lth level if for the maximum width $w_{\max}$ of the boxes on the list L before the actual splitting the following holds:

$$s^{-(l+1)}w(X) < w_{\max} \leqslant s^{-l}w(X)\,. \tag{3}$$

This notion simply follows the way how multisplitting realizes the subdivision. The meaning of it is that if X is an s-quasi hypercube, then just after the 0th level of iteration cycles the subintervals on list L are again congruent s-quasi hypercubes with a width of $w(X)/s$. This property always recurs when entering a new level as a consequence of the interval selection strategy of Hansen, i.e., the Hansen and multisplitting algorithms. The levels of the algorithm are indeed ordered classes over the iteration cycles, as the following remark formulates it.

REMARK. If k and $k+1$ are iteration cycles and k is on the lth level then $k+1$ is either on level l or on level $l+1$.

As Step 4 of the model algorithm is missing in the multisplitting algorithm and since we assume the list L to have infinite capacity, all new subintervals are to be found on our list. The next lemma shows how our list grows.

LEMMA 1. *When beginning with the iteration cycles of level l of the multisplitting algorithm with s-multisplitting, for the number of boxes N_l on the list L the following holds:*

$$N_l = s^{nl} - 1 \,. \tag{4}$$

Proof. When entering a new level, all boxes waiting for splitting are congruent, and thus, their widths are the same, $w_{\max}$. The iteration cycles are finishing level $l-1$ if the last subbox having $w_{\max}$ width is split into s parts. Then the new maximum width is $w_{\max} := w_{\max\_\text{old}}/s$. During this process each box is obviously subdivided into s^n subboxes, hence for all $l \in N$, $N_l = s^n(N_{l-1} + 1)$ holds. Since $N_0 = 0$, namely only the current interval $A = X$ exists before the level 0 iterations, the assertion of the lemma follows. □

The next question arising is how many iteration cycles belong to a level. Some basic steps of the multisplitting method are executed only once in each iteration and thus it is important to be able to calculate the number of iteration cycles that have to be done to enter a new level.

THEOREM 5. *The number of iteration cycles belonging to the lth level is*

$$k_l = \frac{s^{nl}(s^n - 1)}{s - 1} \,. \tag{5}$$

Proof. Each box on the list L and the current box A has to be split into uniform subboxes. Since also at the beginning of a level we have uniform boxes, let us first consider only one of them. Because this is an s-quasi hypercube due to our assumption, first it is sliced through one coordinate direction, then the obtained subboxes through another direction perpendicular to the previous, and so on. At each such stage (or relating to each direction) the number of subboxes to be sliced is s-times greater than t was at the previous stage. The number of stages equals n, i.e., the dimension of the problem. Hence to split one box into uniform subboxes of the next level we need

$$\sum_{i=1}^{n} s^{i-1} = \frac{s^n - 1}{s - 1} \tag{6}$$

iteration cycles.

Having now $N_l + 1 = s^{nl}$ uniform boxes at the beginning (see (4)), multiplying it by the result of (6), the formula for k_l follows. □

Now we know how many iteration cycles belong to a particular level. To be able to calculate the number of iteration cycles which has to be done to reach a certain resolution, first we need to know which iteration cycles are involved in the lth level. The next theorem gives an answer to this question.

THEOREM 6. *Iteration cycle k is on the lth level if and only if*

$$\frac{s^{nl}-1}{s-1}+1 \leqslant k \leqslant \frac{s^{n(l+1)}-1}{s-1}. \tag{7}$$

Proof. Inequalities (7) come directly from (5) of Theorem 5. That theorem states that at the lth level there are exactly k_l iteration cycles. Then summing the $k_l = \frac{s^{nl}(s^n-1)}{s-1}$ terms we have for $l \geqslant 1$

$$\sum_{i=0}^{l-1} k_i = \sum_{i=0}^{l-1} \frac{s^{ni}(s^n-1)}{s-1} = \frac{s^n-1}{s-1} \sum_{i=0}^{l-1} s^{ni} = \frac{s^{nl}-1}{s-1}. \tag{8}$$

Thus, up to the last iteration of level number $l-1$ the whole number of iteration cycles executed is delivered by (8). The next step already belongs to level l resulting in the first inequality of (7).

The lst step of level l can be calculated in a similar way summing to l instead of $l-1$ in (8). That gives us he second inequality of the assertion. □

Now we have characterized the quite natural notion of iteration levels and are nearly ready to give an upper bound on the convergence speed of the multisplitting algorithm in the general case. For the proof of the theorem for the worst case we first cite an important lemma [7]:

LEMMA 2. *If F is an α-convergent inclusion function of f over X then for any* $Y \in \mathbb{I}(X)$

$$\mathrm{lb} f(X) - \mathrm{lb} F(y) = f^* - \mathrm{lb} F(Y) \leqslant c w^{\alpha}(Y) \tag{9}$$

holds where c is the positive constant from the α-convergence definition.

Now we can state the worst case convergence speed for the multisplitting algorithm.

THEOREM 7. *If F is an α-convergent inclusion function of f over X then*

$$\mathrm{lb} f(X) - \min_{Y \in L \cup A} \mathrm{lb} F(Y) \leqslant c w^{\alpha}(X) s^{\alpha} (k(s-1)+1)^{-\alpha/n} \tag{10}$$

holds for the worst case, where c is the smallest positive constant with which the

α-convergence is valid, and L the list of the kth iteration of the multisplitting algorithm.

Proof. From (9) of Lemma 2 it follows that

$$\mathrm{lb}f(X) - \min_{Y \in L \cup A} \mathrm{lb}F(Y) \leqslant cw^{\alpha}(Y^*)\,, \tag{11}$$

where Y^* denotes the interval where the minimum $\min_{Y \in L \cup A} \mathrm{lb}F(Y)$ is reached. If iteration cycle k is on the lth level then from (3) of Definition 3

$$(\forall Y \in L \cup A) \qquad w(Y) \leqslant s^{-l} w(X) \tag{12}$$

follows for any Y subinterval of the kth iteration cycle.

On the other hand, we can reform the second inequality of (7) of Theorem 6:

$$k \leqslant \frac{s^{n(l+1)} - 1}{s - 1}$$

to

$$(k(s-1)+1)^{1/n} s^{l} s\,,$$

and then

$$s^{-l} \leqslant s(k(s-1)+1)^{-1/n}\,. \tag{13}$$

Applying now (13) to (12) and the result to (11), we have exactly what had to be proved. □

This result means that multisplitting influences the theoretical convergence speed. In spite of the $O(k^{-\alpha/n})$ worst case for the Hansen algorithm (see Theorem 3) the presence of the parameter s modifies the worst case to $O(s^{\alpha(n-1)/n} k^{-\alpha/n})$. The new term $s^{\alpha(n-1/n)}$ indicate that in spite of the promising example, multisection may result in a worse efficiency. Though for the same k the upper bound becomes higher and hence worse with increasing s, more information can be gained in a single iteration cycle. Notice that (10) gives exactly (2) for the case $s = 2$. It is worth to remark, that for very high s we cannot complete the first iteration cycle due to memory shortage, and thus after a large number of function evaluations nothing can be said about the optimum. According to Theorem 7, multisplitting does not change the worst case convergence speed for one-dimensional problems.

A more palpable characterization comes directly from the proof of Theorem 7. If we substitute (12) into (11), we have a similar formula for the worst case as in (10) but with parameter l:

$$\mathrm{lb}f(X) - \mathrm{lb}F(A) \leqslant cw^{\alpha}(X) s^{-l\alpha}\,. \tag{14}$$

It is clear that the higher level the algorithms is, the better the optimum f^* is approached by the inclusion function. Inequality (14) describes the exponential nature of this relation. However, the same level can mean different states referring

to the subdivision, i.e., for a greater s the same l attains a finer actual resolution of the domain X but needs more iteration cycles.

At the same time, a single level consists of several iterations. The best case occurs if the considered iteration is the first of its level:

THEOREM 8. *Let F be an α-convergent inclusion function of f over X and let k be the first iteration of some l level. Then the following inequality holds for the best case:*

$$\mathrm{lb}f(X) - \min_{Y \in L \cup A} \mathrm{lb}F(Y) \leqslant cw^{\alpha}(X)((k-1)(s-1)+1)^{-\alpha/n}. \tag{15}$$

Proof. Since k means here the first iteration of level l, the first inequality of (7) holds as an equality. Rearranging this equation we get

$$s^{-l} = ((k-1)(s-1)+1)^{-1/n}. \tag{16}$$

Substituting it into (12), we obtain $w(Y) \leqslant w(X)((k-1)(s-1)+1)$. Now the latter and (11) provide

$$\mathrm{lb}f(X) - \min_{Y \in L \cup A} \mathrm{lb}F(Y) \leqslant cw^{\alpha}(X)((k-1)(s-1)+1)^{-\alpha/n},$$

the statement of Theorem 8. □

In contrast to the result of Theorem 8 on a general upper bound on $\mathrm{lb}f(X) - \min_{Y \in L \cup A} \mathrm{lb}F(Y)$, increasing s means a decrease, i.e. an improvement of this upper bound for the first iteration cycles of the levels.

THEOREM 9. *If we increase the number of intervals split from s to ps ($p > 1$, ps is an integer) of the multisplitting algorithm, then the necessary levels to achieve the same resolution (the maximal width of the boxes in L) of X decreases by a factor of β: $\beta l(s) = l(ps)$, where*

$$\beta \geqslant \left(1 + \frac{\log p}{\log s}\right)^{-1}. \tag{17}$$

The number of iterations changes by a factor of γ: $\gamma k(s) = k(ps)$ at the same time, where

$$\gamma \geqslant \frac{s-1}{ps-1}. \tag{18}$$

Proof. The multisplitting algorithm does not contain any accelerating devices (Step 4) and subdivides the widest box on list L, hence the length of L characterizes the resolution of X. But this length can be determined using Lemma 1. Using s-splitting, the list length is $s^{nl} - 1$ when entering level l, while it is $(ps)^{n\beta l} - 1$ for ps-splitting. Due to our assumption

$$s^{nl} \leqslant (ps)^{n\beta l}, \tag{19}$$

implying

$$n\beta l \geqslant \log_{ps} s^{nl} = nl \frac{\log s}{\log ps} = \frac{nl}{1 + \frac{\log p}{\log s}},$$

what is equivalent to the stated inequality (17). Note that the same result can also be achieved using Definition 3 instead of list lengths.

Because of (8) of the proof of Theorem 6, the number of iterations to be done until reaching the first iteration of level l is

$$k = \frac{s^{nl} - 1}{s - 1}. \tag{20}$$

The corresponding formula for the ps-splitting is

$$\gamma k = \frac{(ps)^{n\beta l} - 1}{ps - 1} \tag{21}$$

to reach level βl. Combining (19) and (21) we obtain

$$\gamma k \geqslant \frac{s^{nl} - 1}{ps - 1}.$$

Dividing it with (20) we get the stated inequality (18) for γ. □

Theorem 9 investigates the case when the multisplitting algorithm completes whole levels. Although (17) is sharp also for the general case in the sense that

$$\beta l = \left\lceil \left(1 + \frac{\log p}{\log s} \right)^{-1} \right\rceil, \tag{22}$$

where $\lceil \cdot \rceil$ denotes the ceiling function, the smallest integer not smaller than the argument. The second statement, (18) of Theorem 9 can deliver in general a rough underestimation and γ can become even greater than 1. If, e.g., an s-splitting terminates finishing a complete level, a ps-splitting can be forced to go through the same number of levels resulting in much more iterations in total. However, some test results have shown a significant decrease in the number of iterations when increasing s from 2 to 3, 4, or 5 in most of the cases.

The aim is first of all to accelerate the Hansen method with the aid of the above outlined modification resulting in the multisplitting algorithm, so let us consider the typically most time consuming parts of it, i.e., those steps of the algorithm which include function calls. Step 1 of the multisplitting algorithm is executed only once, so it should not be considered. Note, however, that local search methods can be used at this step to determine a good upper bound on the global solution of (1) for the cut-off test. Step 2 involves a calculation for determining the splitting direction. This can also mean some inclusion function calls for the objective function and its derivatives (for details see subsection 1.4 and [9]). When implementing the

multisplitting algorithm, Step 4 is usually executed together with Step 3 or Step 5. In the former case certain elements are not even entered to list L, in the latter case a current box is thrown away on the spot if it cannot contain any global minimizers and a new one is chosen. Both of these versions need additional function calls, however, we shall investigate the more widely used former one. Step 5 itself does not need any function calls since the multisplitting algorithm like the Hansen method manages an ordered list with the simple first-in-first-out principle. The termination condition check in Step 6 may require extra function calls, as well.

Summarizing these results we get the following values for the number of inclusion function calls required by the particular iteration cycles of the multisplitting algorithm:

Step 1. –
Step 2. C_1
Step 3. $C_2 s$
Step 4. (included in Step 3)
Step 5. –
Step 6. C_3
Step 7. –

Note that the costs C_i do not depend on s. Hence, the total cost of function calls is

$$C = C_1 + C_2 s + C_3 . \tag{23}$$

Thus, a single iteration cycle costs $O(s)$ function calls. Hence the dependence of the number of function calls on the number of iteration cycles is linear and the magnitude of the convergence speed is the same for the number of function calls as for the number of iteration cycles stated in (10) of Theorem 7.

However, interval subdivision methods are in practice almost never used without accelerating devices. Although they have been used widely for a long time, the investigation of the theoretical effect of these modifications is a difficult problem. The next section provides practical and theoretical results on their effects on the convergence speed.

3. The Accelerating Devices

The generally used accelerating devices are very useful in most cases but it is difficult to treat their effects theoretically when making a worst case analysis. In fact, they do not improve the worst case convergence speed, yet it is worth to involve them into our investigations since for a wide problem class they can improve convergence speed by several magnitudes. Let us utilize their effects of shortening the list L by discarding certain elements at each iteration cycle, regardless of the methods they use to obtain this.

Table 1. The η values, the relative number of subintervals remained after the use of an acceleration test, in percentage for the monotonicity, concavity and cut-off tests on a wide set of global optimization problems according to the levels of nodes in the search tree

Level	Monotonicity t.			Concavity t.			Cut-off t.		
	min.	aver.	max.	min.	aver.	max.	min.	aver.	max.
0	50	91	100	100	100	100	100	100	100
1	3	52	100	97	100	100	3	53	100
2	6	44	83	56	90	100	3	44	81
3	12	31	83	83	95	100	3	34	81
4	12	32	48	88	96	100	3	26	56
5	12	29	46	–	–	–	2	24	45
6	8	27	44	–	–	–	3	18	38
7	7	25	46	–	–	–	3	17	34
8	11	24	43	–	–	–	3	16	34
9	12	28	38	–	–	–	3	17	26
10	11	22	39	–	–	–	3	17	31

Recalling the notion of levels, when entering a new level with the multisplitting algorithm, all $N_l + 1$ boxes waiting for subdivision are congruent. We now assume that at each iteration level at least a certain $1 - \eta$ $(0 < \eta \leqslant 1)$ proportion of boxes is discarded from the list with the aid of a set of accelerating devices, hence the number of elements to be processed can be less than $N_l + 1$ at every level. To demonstrate the viability of this assumption, Table 1 contains test results on typical values of η in practice.

3.1. THE η-ACCELERATED ALGORITHM

To check how realistic this assumption is, we have completed a computational study. The program visited each node of the search tree and checked how many subintervals remained after the use of a given acceleration device (e.g. the cut-off test). This experiment was repeated for each global optimization problem of the later efficiency test (for details see subsection 2.1 in [16]). Table 1 contains the average, the minimum and the maximum of the obtained empirical η values in percentages.

The studied acceleration devices show two phases regarding the achieved deletion rates. In the first 1-2 levels they usually cannot delete a substantial amount of boxes. According to the rounded figures in Table 1, the concavity test e.g. basically could not delete subintervals during the first two search levels. In the subsequent second phase, the average η values improve gradually, and they seem to stabilize around specific values. This trend is also followed by the respective minimal and maximal η values – although they represent sometimes only a single test problem with special characteristic. This is also the reason why the concavity test could not delete any subintervals in the worst case.

Both the monotonicity and the cut-off tests are quite effective. The figures in

Table 1 are in a nice accordance with earlier reports [8, 12], although the optimization algorithms did not check the search tree uniformly. These acceleration devices ensure substantial deletion rates. The concavity test is less effective, so it can be used better as a complementary measure, and in the case if the inclusion of the Hessian is readily available. The missing deletion rates after level 4 for the concavity test are due to memory shortage. Although no $\eta < 1$ value fits each level and each test problem, the subsequent theoretical analysis describes well the general behavior of such accelerated optimization algorithms.

3.2. THEORETICAL CONVERGENCE RESULTS

For the theoretical analysis only the second phase is of importance with stabilizing η values, thus, Step 1 of the multisplitting algorithm also has to be changed. We assume that the first phase is over, there remained $N_0 + 1 > 0$ congruent boxes which are non-overlapping subintervals of X and their union contains all global minimizers. Box A is one of them which is picked up from L. The remaining steps of the algorithm are left untouched. We shall call this algorithm the *η-accelerated algorithm*.

After accomplishing a level of iteration cycles a single box leaves at most $\lfloor \eta s^n \rfloor$ successors on the list instead of having all s^n subboxes stored (here $\lfloor \cdot \rfloor$ stands for the integer part of the argument or the floor function). In general, the next lemma can be stated.

LEMMA 3. *When entering level l, for the number of boxes N_l on the list L the following holds:*

$$N_l \leqslant \lfloor (\eta s^n)^l \rfloor (N_0 + 1) - 1 \,. \tag{24}$$

Proof. During the iteration cycles of a single level, each remained box when this level was entered is subdivided into s^n congruent subboxes. At every iteration cycle of that level some subboxes are deleted so that only at most 100η percentage of them remain by the end of that level. Thus, after completing all iteration cycles of the level, at most ηs^n successors of each box are left, i.e., $N_l \leqslant \lfloor \eta s^n \rfloor (N_{l-1} + 1) - 1$ holds for all $l \in N$. Having $N_0 + 1$ boxes at the beginning, the formula can be expressed explicitly delivering (24). □

This result is very similar to Lemma 1. To be able to state the number of iteration cycles belonging to a single particular level, further circumstances have to be clarified. It is not the same whether the η-accelerating device deletes a few large boxes at the beginning of a level leaving only at most $\lfloor \eta s^n \rfloor^l (N_0 + 1)$ intervals at the end of that level or discards plenty of smaller boxes only already nearing the end of that level. In the former case the number of iteration cycles of that level can be small while in the latter case it can get rather large. The next lemma characterizes

the number of iteration cycles belonging to a level in the best and worst cases, applying an η-accelerating device.

LEMMA 4. *1. For the number of iteration cycles belonging to the lth level of the η-accelerated algorithm the following holds in the best case:*

$$k_l \leqslant \frac{s^n - 1}{s - 1} \lceil \eta \lfloor (\eta s^n)^l (N_0 + 1) \rfloor \rceil. \tag{25}$$

2. In the worst case for the k_l number of iteration cycles belonging to the lth level of the η-accelerated algorithm the following holds:

$$k_l \leqslant \frac{s^n - 1}{s - 1} \lfloor (\eta^n)^l (N_0 + 1) \rfloor. \tag{26}$$

Proof. 1. The best case occurs if $n > 2$, and $\lfloor (1 - \eta)(N_l + 1) \rfloor$ boxes have been discarded during the first iterations of the lth level in full from the list L, without any subdivisions saving

$$\frac{s^n - 1}{s - 1} \lfloor (1 - \eta)(N_l + 1) \rfloor$$

iteration cycles for level l. In the second group of iteration cycles of level l at least further $\lfloor s \operatorname{frac}((1 - \eta)(N_l + 1)) \rfloor$ subboxes of width $s^{-l} w(X)$ are deleted saving

$$\frac{s^{n-1} - 1}{s - 1} \lfloor s \operatorname{frac}((1 - \eta)(N_l + 1)) \rfloor$$

iteration cycles, and then a few further subboxes that have widths of $s^{-l} w(X)$ and $s^{-(l+1)} w(X)$. Here $\operatorname{frac}(x)$ stands for $x - \lfloor x \rfloor$. The last mentioned boxes can be neglected when an upper bound on k_l is calculated. The resulting list of an example 2-dimensional problem after this process is shown in Figure 3, and the deleted boxes are denoted by shading.

Thus, the number of iteration cycles k_l needed is at most

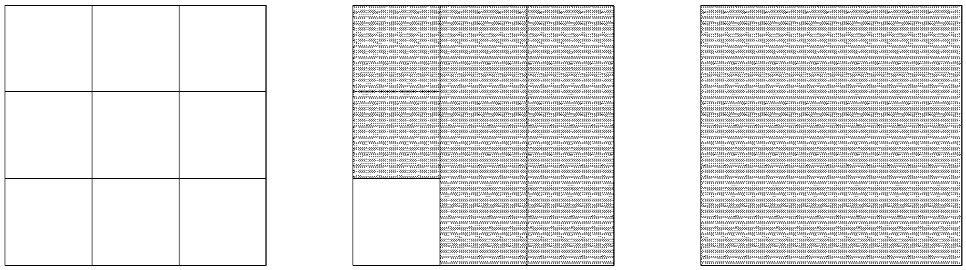

Figure 3. An example outcome of η-acceleration: it is the best case, when the least iteration cycles are made on level 0. The large squares are the 3 starting intervals ($N_0 = 2$), and the deleted subintervals are indicated by shading. The respective parameters were: $s = 3$, $n = 2$, and $\eta = 0.4$.

$$\frac{s^n-1}{s-1}-\frac{s^n-1}{s-1}\lfloor(1-\eta)(N_l+1)\rfloor-\frac{s^{n-1}-1}{s-1}\lfloor s\ \mathrm{frac}((1-\eta)(N_l)+1)\rfloor, \quad (27)$$

and the reformulation gives

$$\begin{aligned} k_l &\leqslant \frac{s^n-1}{s-1}\lfloor\eta(N_l+1)\rfloor+\frac{s^{n-1}-1}{s-1}\lceil s\ \mathrm{frac}((1-\eta)(N_l+1))\rceil \\ &\leqslant \frac{s^n-1}{s-1}\lceil\eta(N_l+1)\rceil. \end{aligned} \quad (28)$$

Substituting $N_l \leqslant \lfloor(\eta s^n)^l\rfloor(N_0+1)-1$ from Lemma 3 into this result we obtain the inequality

$$k_l \leqslant \frac{s^n-1}{s-1}\lceil\eta\lfloor(\eta s^n)^l(N_0+1)\rfloor\rceil,$$

which was to be proved. As we have mentioned, there can be more smaller terms calculated determining a sharper bound for certain cases.

2. In the worst case the remaining boxes are deleted not at the beginning but at the end of that level. Thus, boxes are only deleted when their width is already $s^{-(l+1)}w(X)$ (see Figure 4). Hence, no iteration cycles can be saved for that level, because splitting these subboxes is the duty of the next level.

Entering level l there are at most $\lfloor(\eta s^n)^l(N_0+1)\rfloor$ boxes waiting for subdivision according to (24), each of which needs $(s^n-1)/(s-1)$ iteration cycles as stated in (6) of Theorem 5 to be split into uniform subboxes. The result obtained for the multisplitting algorithm is obviously applicable for the present worst case study. Multiplying these two terms provides the assertion. Also this bound is sharp. □

REMARK. For k_l, the number of iteration cycles belonging to the lth level of the η-accelerated algorithm in the best case the inequality

$$k_l \leqslant \frac{s^n-1}{s-1}\lceil\eta(N_l+1)\rceil-\frac{s^{n-1}-1}{s-1}\lfloor s\ \mathrm{frac}((1-\eta)(N_l+1))\rfloor \quad (29)$$

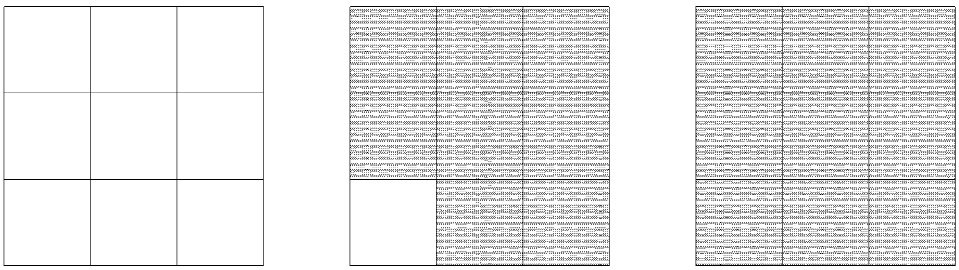

Figure 4. An example outcome of η-acceleration: it is the worst case, when the most iteration cycles are made on level 0. The large squares are the 3 starting intervals ($N_0 = 2$), and the deleted subintervals are indicated by shading. The respective parameters were: $s = 3$, $n = 2$, and $\eta = 0.4$.

holds if $n \leqslant 2$, and for $n \leqslant 2$ no better upper bound can be given utilizing only the η-assumption.

This follows immediately from (27) of the proof of the statement (25) of Lemma 4. For instance, let us consider the example given in Figure 3. The total number of iteration steps of level 0 is 12 in the unaccelerated case, but only 6 with an 0.4-acceleration device. The bound calculated with the aid of (25) is 8, which is correct, but not tight. On the other hand, (29) of Remark provides the correct value

$$4\lceil 1.2 \rceil - \lfloor 3 \operatorname{frac}(1.8) \rfloor = 6 .$$

Of course, the formula can be extended with several terms obtaining sharp bounds for different dimensions n, but for our further investigations this direction is out of interest.

The important point, as it was in the general case, is to characterize the steps belonging to a certain level. The following theorem gives a necessary condition for an iteration cycle to be on the lth level in the worst case.

THEOREM 10. *If the kth iteration cycle is on the lth level then*

1. in the $\lceil \eta s^n \rceil > 1$ *case*

$$k \leqslant \frac{s^n - 1}{s - 1} \frac{\lceil \eta s^n \rceil^{l+1} - 1}{\lceil \eta s^n \rceil - 1} (N_0 + 1) \tag{30}$$

holds while

2. when $\lceil \eta s^n \rceil = 1$ *then*

$$k \leqslant \frac{s^n - 1}{s - 1} (l + 1)(N_0 + 1) . \tag{31}$$

Proof. Lemma 4 provides an upper bound on the number k_l of iteration cycles of the lth level. If $\lceil \eta s^n \rceil > 1$ then summing the k_i terms we get

$$\sum_{i=0}^{l} k_i \leqslant \sum_{i=0}^{l} \frac{s^n - 1}{s - 1} \lfloor (\eta s^n)^i (N_0 + 1) \rfloor \tag{32}$$

$$\leqslant \frac{s^n - 1}{s - 1} (N_0 + 1) \sum_{i=0}^{l} \lceil \eta s^n \rceil^i = \frac{s^n - 1}{s - 1} \frac{\lceil \eta s^n \rceil^{l+1} - 1}{\lceil \eta s^n \rceil - 1} (N_0 + 1) , \tag{33}$$

and that proves assertion 1.

If $\lceil \eta s^n \rceil = 1$ then $\Sigma_{i=0}^{l} \lceil \eta s^n \rceil^i = l + 1$, and inequalities (32) and (33) provide the better upper bound

$$\sum_{i=0}^{l} k_i \leqslant \frac{s^n - 1}{s - 1} (l + 1)(N_0 + 1) , \tag{34}$$

proving assertion 2. □

We cannot give a lower bound for the iteration cycles of level l since our

assumption on the subboxes remaining after each step is only an over-estimate. On the other hand, sharpening our assumption would turn the theoretical results into unrealistic. However, (30) and (31) of Theorem 10 allow us to prove the next result on the worst case convergence speed of the η-accelerated algorithm.

THEOREM 11. *Let F be an α-convergent inclusion function of f over X and let us apply the η-accelerated algorithm to solve (1). Moreover, let the iteration cycle k be on level l at the moment of investigation.*

1. If $\lceil \eta s^n \rceil > 1$ is true then

$$\mathrm{lb} f(X) = \min_{Y \in L \cup A} \mathrm{lb} F(Y) = O(\eta^{\alpha l/n} s^{\alpha(n-1)/n} k^{-\alpha/n}) \tag{35}$$

holds.

2. In the $\lceil \eta s^n \rceil = 1$ case the convergence can be characterized by:

$$\mathrm{lb} f(X) - \min_{Y \in L \cup A} \mathrm{lb} F(Y) = O(s^{-\alpha l/n} k^{-\alpha l/n} l^{\alpha l/n}) = O\left(\left(\frac{l}{sk}\right)^{\alpha l/n}\right). \tag{36}$$

Proof. 1. Let us assume that

$$\lceil \eta s^n \rceil \geqslant 2 . \tag{37}$$

Since $\lceil \eta s^n \rceil$ is positive and the $\lceil \eta s^n \rceil = 1$ case is investigated separately, condition (37) is always fulfilled. Thus, inequality (30) of Theorem 10 is valid and it can easily be rearranged based on the natural assumptions of $s > 1$ and $N_0 + 1 > 0$ to

$$\frac{k(s-1)(\eta s^n - 1)}{(N_0+1)(s^n-1)} + 1 \leqslant \frac{k(s-1)(\lceil \eta s^n \rceil - 1)}{(N_0+1)(s^n-1)} + 1 \leqslant \lceil \eta s^n \rceil^{l+1} . \tag{38}$$

Subsequently, we give an upper bound on $\lceil \eta s^n \rceil^{l+1}$. Obviously, $\lceil \eta s^n \rceil < \eta s^n + 1$, and $\eta s^n > 1$ holds because of condition (37). Hence, $\eta s^n + 1 < 2\eta s^n$. This provides the following estimation:

$$\lceil \eta s^n \rceil^{l+1} < (2\eta s^n)^{l+1} = (2\eta)^{l+1} (s^l s)^n . \tag{39}$$

Inequalities (38) and (39) together give an upper bound for s^{-l}:

$$s^{-l} < s \left(\frac{\frac{k(s-1)(\eta s^n - 1)}{(N_0+1)(s^n-1)} + 1}{(2\eta)^{l+1}} \right)^{-\frac{1}{n}} . \tag{40}$$

The η-accelerated algorithm is a Hansen method concerning the way it selects the current box, hence, from Definition 3

$$w(A) = w_{\max} \leqslant s^{-l} w(X) \tag{41}$$

holds for all current boxes A of the lth level. Because all boxes Y stored in the list L have less or equal widths as the current box, for the interval Y^* where the minimum $\min_{Y \in L \cup A} \mathrm{lb} F(Y)$ is obtained the inequality

$$w(Y^*) \leqslant s^{-l} w(X) \tag{42}$$

also holds.

Applying (42) to Lemma 2 we have

$$\mathrm{lb} f(X) - \min_{Y \in L \cup A} \mathrm{lb} F(Y) \leqslant c w^{\alpha}(Y^*) \leqslant c(s^{-l} w(X))^{\alpha} . \tag{43}$$

Let us continue the inequalities (43) substituting (40) into it:

$$c(s^{-l} w(X))^{\alpha} < c\left(s w(X) \left(\frac{\frac{k(s-1)(\eta s^n - 1)}{(N_0 + 1)(s^n - 1)} + 1}{(2\eta)^{k+1}} \right)^{-\frac{1}{n}} \right)^{\alpha} . \tag{44}$$

Checking the terms' magnitudes assertion 1 follows.

2. In the $\lceil \eta s^n \rceil = 1$ case the inequality (31) of Theorem 10 holds. Rearranging (31) we obtain:

$$s^{-l} \leqslant \left(\frac{k(s-1)}{(l+1)(N_0 + 1)} + 1 \right)^{-\frac{l}{n}} . \tag{45}$$

Based on the argumentation of the proof of assertion 1 above, (45) can be substituted into (43) delivering

$$\mathrm{lb} f(X) - \min_{Y \in L \cup A} \mathrm{lb} F(Y) \leqslant c \left(\frac{k(s-1)}{(l+1)(N_0 + 1)} + 1 \right)^{-\frac{\alpha l}{n}} w(X)^{\alpha} , \tag{46}$$

and that implies (36). □

Note that in (10) of Theorem 7, which corresponds to the unaccelerated multisplitting algorithm the magnitudes of the terms s and k are the same as in the accelerated case as stated in (35) of Theorem 11. The only difference in non-constant terms is the presence of η. This coefficient expresses the weakness of the η-accelerating device. Thus, the greater this η is the worse approximation for the optimum can be achieved using a fixed amount of computation. The $\lceil \eta s^n \rceil = 1$ case is very special and quite rare. The bound given in (36) of Theorem 11 is always tighter than the bound of (35). The latter statement also means that the bound of (35) can always be used, and the special case has only been investigated to be exhaustive. Summarizing the consequences of Theorem 11, the convergence speed depends on η as $\eta^{\alpha l/n}$. In other words, for fixed α-convergence rate and dimension n, better acceleration devices result in polynomially better convergence speed: the larger level we are, the quicker the convergence speed. This time no best case convergence speed can be given due to the definition of η (see also the comment after the proof of Theorem 10).

All theoretical convergence results given here for multisplitting or η-accelerated algorithms make only sense if the notion of iteration levels is valid for the method in question. Any attempt to translate the train of thought directly to the widely used Moore-Skelboe methods would fundamentally fail. The reason for this is again to be

found in how the Hansen and the Moore-Skelboe methods search the tree of subdivisions to tighten the inclusion of the global optimum value. While the Hansen methods perform an accurate breadth-first search, a Moore-Skelboe method can go deep down the tree and then reach out for a box being much higher, depending on the particular problem to be solved. This is the reason why the otherwise easily available inclusion isotonicity property of the inclusion function is required for Moore-Skelboe methods to be acceptably effective. The theoretical investigation of Moore-Skelboe methods needs therefore a completely different point of view, and such a study exceeds the range of this paper.

4. Summary and Conclusions

Compared to stochastic methods, interval methods for global optimization are able to provide solutions of guaranteed reliability – at the costs of sometimes substantially higher computational and space complexity. The present study aimed at investigating the possibilities of improving the efficiency while keeping the reliability. A new subdivision technique is applied where the traditional bisection step is substituted by the subdivision of the current interval into many subintervals in a single iteration step. For the multisplitting algorithm the number of iteration cycles belonging to a level of subintervals and the serial numbers of such iteration cycles were determined. On this basis, the speed of the convergence to the global minimum value was given for the best and the worst cases. According to these convergence speed bounds, the worst case speed is the best for bisection, while the best case speed improves as s, the number of subintervals generated in a single iteration step increases. The effects of the accelerating devices on the convergence speed are specified, and it turned out that better acceleration devices result in polynomially better convergence speed, and on the larger level we are, the quicker the convergence speed.

Acknowledgement

The authors are grateful for the useful suggestions of the two anonymous referees.

References

1. Alefeld, G. and Herzberger, J. (1983), *Introduction to Interval Computations*. Academic Press, New York.
2. Berner, S. (1995), *Ein paralleles Verfahren zur verifizierten globalen Optimierung*. Dissertation, Universität Wuppertal.
3. Berner, S. (1966), New results on verified global optimization, *Computing* 57: 323–343.
4. Casado, L.G., García, I. and Csendes, T., Adaptive Multisection in Interval Methods for Global Optimization, submitted for publication
(available at http://www.inf.u-szeged.hu/~csendes/publ.html).

5. Casado, L.G., García, I. and Csendes, T., A Heuristic Rejection Criterion in Interval Global Optimization Algorithms, submitted for publication (available at http://www.inf.u-szeged.hu/~csendes/publ.html).
6. Csallner, A.E. (1993), Global optimization in separation network synthesis, *Hungarian J. Industrial Chemistry* 21: 303–308.
7. Csallner, A.E. and Csendes, T. (1996), On the convergence speed of interval methods for global optimization, *Computers, Mathematics and Applications* 31: 173–178.
8. Csendes, T. and Pintér, J. (1993), The impact of accelerating tools on the interval subdivision algorithm for global otimization, *European J. of Operational Research* 65: 314–320.
9. Csendes, T. and Ratz, D. (1997), Subdivision direction selection in interval methods for global optimization, *SIAM J. Numerical Analysis* 34: 922–938.
10. Hansen, E. (1992), *Global optimization using interval analysis*. Marcel Dekker, New York.
11. Hansen, P., Jaumard, B. and Xiong, J. (1994), Cord-Slope Form of Taylor's Expansion in Univariate Global Optimization, *J. Optimization Theory and Applications* 80: 441–464.
12. Ichida, K. and Fujii, Y. (1979), An interval arithmetic method for global otimization. *Computing* 23: 85–97.
13. Kearfott, R.B. (1996), Test results for an interval branch and bound algorithm for equality-constrained optimization. In: Floudas, C.A. and Pardalos, P.M. (eds.), *State of the Art in Global Optimization*. Kluwer, Dordrecht, 181–199.
14. Kearfott, R.B. (1996), *Rigorous global search: continuous problems*. Kluwer, Dordrecht.
15. Krawczyk, R. and Neumaier, A. (1985), Interval Slopes for Rational Functions and Associated Centered Forms, *SIAM J. Numerical Analysis* 22: 604–616.
16. Markót, M.Cs., Csendes, T. and Csallner, A.E., Multisection in Interval Branch-and-Bound Methods for Global Optimization II. Numerical Tests. Submitted for publication (available at http://www.inf.u-szeged.hu/~csendes/publ.html).
17. Ratschek, H. and Rokne, J. (1988), *New Computer Methods for Global Optimization*. Ellis Horwood, Chichester.
18. Ratschek, H. and Rokne, J. (1993), Interval Methods, In: Horst R. and Pardalos P.M. (eds.), *Handbook of Global Optimiation*, Kluwer, Dordrecht, 751–828.
19. Ratz, D. (1992), *Automatische Ergebnisverifikation bei globalen Optimierungsproblemen*. Dissertation, Univesität Karlsruhe.
20. Ratz, D. and Csendes, T. (1995), On the selection of subdivision directions in interval branch-and-bound methods for global optimization, *J. Global Optimization* 7, 183–207.