Mathematical Programming manuscript No.
(will be inserted by the editor)

M.Cs. Markót · J. Fernández · L.G. Casado · T. Csendes

New interval methods for constrained global optimization

Received: date / Revised version: date

Abstract. Interval analysis is a powerful tool which allows to design branch-and-bound algorithms able to solve many global optimization problems. In this paper we present new adaptive multisection rules which enable the algorithm to choose the proper multisection type depending on simple heuristic decision rules. Moreover, for the selection of the next box to be subdivided, we investigate new criteria. Both the adaptive multisection and the subinterval selection rules seem to be specially suitable for being used in inequality constrained global optimization problems. The usefulness of these new techniques is shown by computational studies.

Keywords: global optimization, inequality constrained problems, interval analysis, adaptive multisection, subinterval selection criterion, computational study.

1. Introduction

Let $\boldsymbol{X}_0 \subseteq \mathbb{R}^n$ be the n-dimensional interval (also called a *box*) where the optimizer points are searched. Let f and g_j, $j = 1, \ldots, r$ be real-valued functions defined on $\boldsymbol{X}_0$. We assume that f and g_j, $j = 1, \ldots r$ are continuous. The *inequality constrained global optimization* (*minimization*) *problem* is usually written in the following form:

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & g_j(x) \le 0, \ j = 1 \ldots, r \\ & x \in \boldsymbol{X}_0. \end{array} \tag{1}$$

Due to the possible existence of local (in contrast to global) minimizer points, (1) is a difficult-to-solve problem. Standard techniques in nonlinear optimization locate usually just local minima. Moreover, there is no local information based criterion for deciding whether a local solution is global. Therefore, conventional optimization methods using derivative information are, in general, not capable of locating or identifying a global minimum point. Due to these difficulties, the

Mihály Csaba Markót: Research Group on Artificial Intelligence of the Hungarian Academy of Sciences, University of Szeged, Hungary, email: markot@inf.u-szeged.hu

José Fernández Hernández: Department of Statistics and Operations Research, University of Murcia, Spain, email: josefdez@um.es

Leocadio González Casado: Department of Computer Architecture and Electronics, University of Almería, Spain, email: leo@ace.ual.es

Tibor Csendes: Institute of Informatics, University of Szeged, Hungary, email: csendes@inf.u-szeged.hu

Mathematics Subject Classification (1991): 65G10, 90C30, 65K05, 90B85

methods devised for solving (1) are quite diverse and different from the standard tools and they depend on the additional properties required on the objective function f and/or on the constraint functions g_j. For an introduction to these methods, the interested reader is referred to [24,25].

One of the most successful methods dealing with (1) use interval analysis in branch-and-bound algorithms. Interval branch-and-bound methods work in a reliable way (usually they apply interval arithmetic [23,28,37] to have inclusion functions for the objective function and for the constraint functions). Basically, they proceed as follows. They start with the initial box, $\boldsymbol{X}_0$. The box considered is either sent to the solution list, it is removed from further consideration by a 'discarding test', or it is split into several subboxes which are considered later. This process is repeated by choosing a new box until no box remains to be considered or until a global minimizer point is found.

In this paper we investigate two parts of the basic branch-and-bound algorithms. On the one hand, we study the subdivision step, especially the number of subboxes into which a given box is split. On the other hand, we investigate the step in which the next box to be processed by the algorithm is chosen. These proved to be important steps in the algorithm since, as the numerical experiments show, if we subdivide a box into a suitable number of subboxes, or if we select the proper box for further subdivision, the speed of the algorithm can be increased substantially.

The paper is organized as follows. In Section 2, we introduce the notation, present a prototype interval global optimization algorithm and discuss the subdivision and subinterval selection methods used in the interval analysis literature. In Section 3 we define an index which measures the feasibility degree of a box and propose a new adaptive multisection rule for inequality constrained problems based on that index. Moreover, we point out how that index can be applied for creating new subinterval selection rules. Some theoretical convergence statements are presented in Section 4. A computational study which shows the efficiency improvements of the new techniques is presented in Section 5. Finally, the last section contains the conclusions and an outline of the directions for future research.

2. Interval Analysis and Global Optimization

In this section, we briefly summarize the fundamental concepts of interval analysis which are needed for this paper. For more details, the interested reader is referred to [23,27,37,39].

In the notation used here, boldface will denote intervals. Brackets "$[\cdot]$" will delimit intervals, while parentheses "$(\cdot)$" vectors and matrices. Underlining will denote lower bounds of intervals and overlining will give the upper bounds. For example, we may have the interval vector $\boldsymbol{X} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)^T$, where $\boldsymbol{X}_i = [\underline{\boldsymbol{X}}_i, \overline{\boldsymbol{X}}_i]$. The *width* of an interval $\boldsymbol{X}$ is denoted by $w(\boldsymbol{X}) = \overline{\boldsymbol{X}} - \underline{\boldsymbol{X}}$, and its midpoint by $m(\boldsymbol{X}) = (\underline{\boldsymbol{X}} + \overline{\boldsymbol{X}})/2$. The width of an interval vector $\boldsymbol{X} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)^T$ is defined as $w(\boldsymbol{X}) = \max\{w(\boldsymbol{X}_i) : i = 1, \ldots, n\}$ and its midpoint as $m(\boldsymbol{X}) = (m(\boldsymbol{X}_1), \ldots, m(\boldsymbol{X}_n))^T$. The set of compact intervals will be denoted by $\mathbb{I}$, and the set of n-dimensional interval vectors, also called *boxes*, by $\mathbb{I}^n$. (If it is not confusing, we sometimes call boxes also as intervals.)

The *interval arithmetic operations* are defined by

$$\boldsymbol{X} * \boldsymbol{Y} = \{x * y : x \in \boldsymbol{X}, y \in \boldsymbol{Y}\} \text{ for } \boldsymbol{X}, \boldsymbol{Y} \in \mathbb{I}, \tag{2}$$

where the symbol $*$ stands for $+, -, \cdot$ and $/$ ($\boldsymbol{X}/\boldsymbol{Y}$ is only defined when $0 \notin \boldsymbol{Y}$). Definition (2) can be fulfilled by simple constructive rules (see [23,27,37, 39]). The algebraic properties of (2) are different from those of real arithmetic operations (for instance, the subtraction and division in $\mathbb{I}$ are not the inverse operations of addition and multiplication, respectively), but the main properties from the operational point of view still hold in a slightly different way, as for instance the so-called subdistributive law,

$$\boldsymbol{X} \cdot (\boldsymbol{Y} + \boldsymbol{Z}) \subseteq \boldsymbol{X} \cdot \boldsymbol{Y} + \boldsymbol{X} \cdot \boldsymbol{Z} \quad \text{for} \quad \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z} \in \mathbb{I},$$

and the *inclusion isotonicity*,

$$\boldsymbol{X} \subseteq \boldsymbol{Y}, \boldsymbol{Z} \subseteq \boldsymbol{T} \Longrightarrow \boldsymbol{X} * \boldsymbol{Z} \subseteq \boldsymbol{Y} * \boldsymbol{T} \text{ (if } \boldsymbol{Y} * \boldsymbol{T} \text{ is defined)} \quad \text{for } \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{T} \in \mathbb{I}.$$

Definition 1. *A function $\boldsymbol{f} : \mathbb{I}^n \to \mathbb{I}$ is said to be an* inclusion function *of $f : \mathbb{R}^n \to \mathbb{R}$ provided*

$$\{f(x) : x \in \boldsymbol{X}\} \subseteq \boldsymbol{f}(\boldsymbol{X})$$

for all boxes $\boldsymbol{X} \subset \mathbb{I}^n$ within the domain of $\boldsymbol{f}$.

Observe that when $\boldsymbol{f}$ is an inclusion function for f then we can directly obtain lower and upper bounds of f over any box $\boldsymbol{X}$ within the domain of f just by taking $\underline{\boldsymbol{f}}(\boldsymbol{X})$ and $\overline{\boldsymbol{f}}(\boldsymbol{X})$, respectively.

For a standard function h (like sin, exp, etc.) predefined in some programming language, it is not too difficult to obtain a good inclusion function $\boldsymbol{h}$, since the monotonicity properties of these functions are well known and then we can take $\boldsymbol{h}(\boldsymbol{X}) = \{h(x) : x \in \boldsymbol{X}\}$ for any $\boldsymbol{X} \in \mathbb{I}$ in the domain of h. For a general function $f(x)$, $x \in \mathbb{R}^n$, the easiest method to obtain an inclusion function is the so-called *natural interval extension*, which is obtained by replacing each occurrence of the variable x with a box including it, $\boldsymbol{X}$, each occurrence of a predeclared function h by its corresponding inclusion function $\boldsymbol{h}$, and the real arithmetic operators by the corresponding interval operators.

Some important concepts for inclusion functions are given in the following definitions:

Definition 2. *The inclusion function $\boldsymbol{f}$ is said to be an* isotone inclusion function *over $\boldsymbol{X}_0$ if for any pair of boxes $\boldsymbol{Y}, \boldsymbol{Z} \subseteq \boldsymbol{X}_0$, $\boldsymbol{Y} \subseteq \boldsymbol{Z}$ implies $\boldsymbol{f}(\boldsymbol{Y}) \subseteq \boldsymbol{f}(\boldsymbol{Z})$.*

Definition 3. *We call the inclusion function $\boldsymbol{f}$ an* α-convergent inclusion function *over $\boldsymbol{X}_0$ if for any box $\boldsymbol{Y} \subseteq \boldsymbol{X}_0$, $w(\boldsymbol{f}(\boldsymbol{Y})) - w(f(\boldsymbol{Y})) \leq K(w(\boldsymbol{Y}))^\alpha$ holds, where α and K are positive constants and $f(\boldsymbol{Y})$ is the range of f over $\boldsymbol{Y}$.*

Definition 4. *We say that the inclusion function* $\boldsymbol{f}$ *has the* zero convergence property*, if* $w(\boldsymbol{f}(\boldsymbol{Z}_i)) \to 0$ *holds for all the* $\{\boldsymbol{Z}_i\}$ *interval sequences for which* $\boldsymbol{Z}_i \subseteq \boldsymbol{X}_0$ *for all* $i = 1, 2, \dots$ *and* $w(\boldsymbol{Z}_i) \to 0$.

The prototype interval branch-and-bound algorithm for solving (1) is as follows.

Prototype algorithm

1. $\boldsymbol{Y} \leftarrow \boldsymbol{X}_0$, initialize the empty list $\mathcal{L}_\mathcal{W}$.
2. Choose coordinate directions for the splitting of $\boldsymbol{Y}$.
3. Split $\boldsymbol{Y}$ normal to the chosen directions, cutting the box a given number of times in each direction. Let $\boldsymbol{Y}_1, \dots, \boldsymbol{Y}_s$ be the subboxes obtained.
4. For $i = 1$ to s do
   - 4.1. Delete $\boldsymbol{Y}_i$ if it can be proven that $\boldsymbol{Y}_i$ contains no optimal solution or diminish $\boldsymbol{Y}_i$ if it can be proven that the respective part of $\boldsymbol{Y}_i$ contains no optimizer point.
   - 4.2. If $\boldsymbol{Y}_i$ is not deleted, then store it (as a whole or diminished) into the working list $\mathcal{L}_\mathcal{W}$.
5. Choose a box from $\mathcal{L}_\mathcal{W}$ and remove it from that list. Let $\boldsymbol{Y}$ denote the chosen box.
6. While the termination criterion does not hold go to Step 2.

Different methods can be derived depending on how the Steps 2, 3, 4, 5, and 6 are executed. A large part of the interval global optimization literature is devoted to step 4.1, i.e., to tests for verifying that a box or a part of a box contains no feasible optimal point. We next briefly discuss some of the most successful ones.

We say that a box $\boldsymbol{Y}$ *certainly satisfies* the constraint $g_j(x) \le 0$ if $\overline{\boldsymbol{g}}_j(\boldsymbol{Y}) \le 0$ and that $\boldsymbol{Y}$ *does certainly not satisfy* it if $\underline{\boldsymbol{g}}_j(\boldsymbol{Y}) > 0$. A box $\boldsymbol{Y} \subseteq \boldsymbol{X}_0$ is said *certainly feasible* if it certainly satisfies all the constraints, *certainly infeasible* if it does certainly not satisfy at least one of the constraints, and *undetermined* otherwise. A box $\boldsymbol{Y} \subseteq \boldsymbol{X}_0$ is said *certainly strictly feasible* if $\overline{\boldsymbol{g}}_j(\boldsymbol{Y}) < 0$, $j = 1, ..., r$. The 'feasibility test' discards boxes that are certainly infeasible. Notice that to apply this test we just need an inclusion function $\boldsymbol{g}_j$ for each of the functions g_j defining the constraints.

Every time a box $\boldsymbol{X}$ is chosen from the list $\mathcal{L}_\mathcal{W}$, and provided that its midpoint c is certainly feasible, we compute $\overline{\boldsymbol{f}}(c)$ (considering c as a point box, i.e. a box with zero width in each component) and update the best upper bound $\tilde{f}$ of the global minimum f^*. The 'midpoint test' discards boxes whose objective function value at all of its points is greater than the current $\tilde{f}$, i.e., a box $\boldsymbol{Y}$ is removed if $\underline{\boldsymbol{f}}(\boldsymbol{Y}) > \tilde{f}$. Also, a new box $\boldsymbol{Y}$ needs only be entered into $\mathcal{L}_\mathcal{W}$ if $\tilde{f} \ge \underline{\boldsymbol{f}}(\boldsymbol{Y})$ is satisfied. We often refer to this test also as 'cut-off test'. The updating of $\tilde{f}$ can also be done as $\tilde{f} := \min\{\tilde{f}, \overline{\boldsymbol{f}}(\boldsymbol{X})\}$, provided that the box $\boldsymbol{X}$ contains at least one feasible point.

Assuming that the objective function f is in addition continuously differentiable, the 'monotonicity test' can be used to decide whether f is strictly monotonous in a certainly strictly feasible box $\boldsymbol{Y} \subseteq \boldsymbol{X}_0$. Then, $\boldsymbol{Y}$ cannot contain a global minimizer in its interior. If $\boldsymbol{\nabla f} = (\boldsymbol{\nabla f}_1, \ldots, \boldsymbol{\nabla f}_n)^T$ is an inclusion function for the gradient ∇f of the objective function, and for a certainly strictly feasible box $\boldsymbol{Y} \subseteq \boldsymbol{X}_0$ we have that $0 \notin \boldsymbol{\nabla f}_i(\boldsymbol{Y})$ for any i, then $\boldsymbol{Y}$ can be deleted or reduced to a respective side. Other discarding tests can be found in [19,22, 34,43].

The working list $\mathcal{L}_\mathcal{W}$ is usually ordered according to a selection criterion used to select the box to be subdivided (Step 5). The selected box is often called the *leading box* of the next iteration step. The two most widely used selection criteria are:

C1: to select the box with the minimal lower bound $\underline{\boldsymbol{f}}(\boldsymbol{Y})$ (known as Moore-Skelboe rule [37]), and

C2: to choose the oldest (i.e. the widest) interval from the list (known as Hansen rule [23]).

These rules determine the theoretic convergence properties of the model algorithm (for details see [8,13,37]). From the computational point of view, rule C1 ensures a quicker algorithm than C2. Furthermore, Berner [1] suggested that the most suitable box to choose from the working list is the one with the smallest lower bound $\underline{\boldsymbol{f}}(\boldsymbol{Y})$, since in this way the boxes rarely will be subdivided unnecessarily.

Casado, Martínez and García [6] and Csendes [11] have recently experimented on unconstrained problems with new interval selection criteria based on variants of the *RejectIndex,*

$$pf^*(\boldsymbol{X}) = \frac{f^* - \underline{\boldsymbol{f}}(\boldsymbol{X})}{w(\boldsymbol{f}(\boldsymbol{X}))},$$

an indicator of the proximity of a box to a minimizer point, and thus, an estimate of the chance a box contains an optimizer point:

C3: As in [11], the box with the maximal $pf^*(\boldsymbol{X})$ value is selected.

C4: As in [6], a hybrid selection rule is proposed. Let N_m be a given constant. From the current set of boxes in a given *iteration level* (i.e. in a given level of the B&B tree generated by the algorithm), the algorithm chooses at most N_m of them, the ones with the largest $pf^*(\boldsymbol{X})$ values. The rest of the boxes still of interest are stored in a secondary list and are processed at the end of the algorithm. The subboxes still of interest generated by the subdivision of the selected boxes will form the new set of boxes for the next iteration level.

Numerical results show that substantial improvements can be achieved with these new selection rules [6,11]. The $pf^*(\boldsymbol{X})$ index has also proved to be a good basis for load balancing in parallel algorithms [3] and for a heuristic rejection criterion solving hard unconstrained optimization problems that have large memory complexity [5].

Concerning the subdivision process (Steps 2 and 3), the classical method is the bisection of the box perpendicular to a direction of maximum width. Nevertheless, neither that selection of the direction nor the number of subboxes generated seems to be the best choice. In recent papers (see [14,15,41,42]) Csendes and Ratz have studied rules for the selection of subdivision directions (Step 2). Each of the rules selects a direction k by using a merit function:

$$k = \min\{j \mid j \in \{1,\ldots,n\} \text{ and } D(j) = \max{}_{i=1}^{n} D(i)\}$$

where $D(i)$ is determined by the given rule. They have empirically proved, using a wide spectrum of unconstrained test problems, that the choice of some rules (in particular, rule B, defined by $D(i) = w(\boldsymbol{\nabla f}_i(\boldsymbol{X}))w(\boldsymbol{X}_i)$, and rule C, defined by $D(i) = w(\boldsymbol{\nabla f}_i(\boldsymbol{X})(\boldsymbol{X}_i - m(\boldsymbol{X}_i))))$ may reduce the CPU time required for solving different problems by around 20% as compared with the classical rule which selects the direction of maximum width ($D(i) = w(\boldsymbol{X}_i)$).

As for the number of subboxes s into which to subdivide a box (Step 3), which is the other main target of the paper, computational studies with unconstrained problems [1,35,41] have shown that it is better to bisect a box always twice (orthogonal to two coordinate directions) than only once in one step, which is the standard case in the literature. Compared to the classical bisection, around 20% of time can be gained on average with the use of the tetrasection. The number of boxes to be processed by the algorithm is related to the kind of subdivision used ([2], Theorem 3.3.1): for an interval branch-and-bound algorithm the use of multisection instead of bisection may, in the best case, save the processing of nearly half of the boxes, and in the worst case, provoke to have to process an arbitrary large number of boxes as compared with bisection.

Unfortunately, for a given problem, and for some boxes bisection may be better than multisection whereas for other boxes the converse may hold. Thus an *adaptive multisection*, as the one presented in [4] by Casado, García and Csendes, may be a better option. This rule is based on experiences that suggest the subdivision of the current box into a larger number of pieces only if it is located in the neighbourhood of a minimizer point. In that paper the *RejectIndex* parameter was used as an estimate of the proximity of a box to a minimizer point: the higher the $pf^*(\boldsymbol{X})$ value, the closer the box $\boldsymbol{X}$ to a minimizer point. The computational results in [4] show that significant improvements can be obtained with the use of adaptive multisection as compared with the static bisection and multisection rules.

In practice, the global minimum value, f^*, is usually not known in advance. Thus, instead of the *RejectIndex*, the estimate

$$p(\hat{f}, \boldsymbol{X}) = \frac{\hat{f} - \underline{\boldsymbol{f}}(\boldsymbol{X})}{w(\boldsymbol{f}(\boldsymbol{X}))},$$

is used, which still keeps the basic properties of $pf^*(\boldsymbol{X})$. $\hat{f}$ is an approximation of the global minimum value f^*.

For unconstrained problems, $p(\tilde{f}, \boldsymbol{X}) \in [0,1]$ for all the boxes $\boldsymbol{X}$ still of interest: all the boxes for which $p(\tilde{f}, \boldsymbol{X}) < 0$ holds are removed by the midpoint test, and $p(\tilde{f}, \boldsymbol{X}) > 1$ cannot hold since we always update $\tilde{f}$ by evaluating $\boldsymbol{f}$ at the midpoint of the current box. Notice that $p(\tilde{f}, \boldsymbol{X}) > 1$ cannot hold either if the updating of $\tilde{f}$ is done with $\overline{\boldsymbol{f}}(\boldsymbol{X})$.

For constrained problems when f^* is approximated by $\tilde{f}$, the $p(\tilde{f}, \boldsymbol{X})$ value of some undetermined boxes may be greater than 1. Notice that if the midpoint of the current box is infeasible, we cannot update $\tilde{f}$ by $\tilde{f} := \min\{\tilde{f}, \overline{\boldsymbol{f}}(m(\boldsymbol{X}))\}$. On the other hand, if $\overline{\boldsymbol{f}}(\boldsymbol{X}) < \tilde{f}$, we can only set $\tilde{f} := \overline{\boldsymbol{f}}(\boldsymbol{X})$ if we can guarantee that $\boldsymbol{X}$ contains at least one feasible point, which may be a difficult task. For the adaptive multisection, a $p(\tilde{f}, \boldsymbol{X})$ value greater than 1 is not a problem: it just means that the box is both promising (it is possibly close to a minimizer point of f) and undetermined. The problem is to decide on the subdivision type to be performed on undetermined boxes. The less feasible the box is, the less worth is the subdivision of the box into many subboxes, since in many cases most of the area covered by the infeasible subboxes could be removed by the feasibility test applied to a bigger subbox, which may avoid the processing of many smaller subboxes. Furthermore, the less feasible the box is, the less likely the box contains a minimizer point. Notice that if $\underline{\boldsymbol{f}}(\boldsymbol{X})$ is very low, but most part of $\boldsymbol{X}$ is infeasible, then $\underline{\boldsymbol{f}}(\boldsymbol{X})$ may be determined by the infeasible points of $\boldsymbol{X}$. Hence, concerning the selection criterion, in that case it would be better to choose another box $\boldsymbol{Y}$ with a higher $\underline{\boldsymbol{f}}(\boldsymbol{Y})$ value but more feasible. By doing this, we may not only choose a more suitable box, but also reduce $\tilde{f}$ quicker (and according to [12] this is important).

3. A feasibility degree index for constrained problems

As we have seen, we need an index which measures the feasibility degree of a box. Next, we present such an index, which is simple to obtain. Let us consider a single constraint, $g_j(x) \leq 0$. The index we propose to measure the feasibility degree of a box $\boldsymbol{X}$ with regard to that constraint is given by:

$$pu_{\boldsymbol{g}_j}(\boldsymbol{X}) = \min\left\{\frac{-\underline{\boldsymbol{g}}_j(\boldsymbol{X})}{w(\boldsymbol{g}_j(\boldsymbol{X}))}, 1\right\}.$$

In the very rare case, when $w(\boldsymbol{g}_j(\boldsymbol{X})) = 0$, the feasibility could be decided directly, and we set $pu_{\boldsymbol{g}_j}$ accordingly to -1 or 1.

Notice that if $pu_{\boldsymbol{g}_j}(\boldsymbol{X}) < 0$ then the box is certainly infeasible, and if $pu_{\boldsymbol{g}_j}(\boldsymbol{X}) = 1$ then $\boldsymbol{X}$ certainly satisfies the constraint. Otherwise, the box is undetermined for that constraint. Here the drawback can be that, due to the overestimation of the inclusions, we can observe an undetermined status also for fully feasible boxes.

Second, let us consider the set of constraints defining the feasible set of (1), $g_1(x) \leq 0, \ldots, g_r(x) \leq 0$. For the boxes which are not certainly infeasible, i.e., for which $pu_{\boldsymbol{g}_j}(\boldsymbol{X}) \geq 0$ for all $j = 1, \ldots, r$, holds, the index is given by

$$pu(\boldsymbol{X}) = \prod_{j=1}^{r} pu_{\boldsymbol{g}_j}(\boldsymbol{X}).$$

We must only define the index for such boxes since certainly infeasible boxes are immediately removed by the algorithm from further consideration. With this definition,

- $pu(\boldsymbol{X}) = 1 \Leftrightarrow \boldsymbol{X}$ is certainly feasible, and
- $pu(\boldsymbol{X}) \in [0, 1) \Leftrightarrow \boldsymbol{X}$ is undetermined.

In the definition of $pu_{\boldsymbol{g}_j}(\boldsymbol{X})$, its maximum value has been bounded by 1, since $-\underline{\boldsymbol{g}}_j(\boldsymbol{X})/w(\boldsymbol{g}_j(\boldsymbol{X})) > 1$ just means that $\boldsymbol{X}$ certainly satisfies that constraint, but the inclusion of such a value in the product $\prod_{j=1}^{r} pu_{\boldsymbol{g}_j}(\boldsymbol{X})$ could lead us to consider an undetermined box $\boldsymbol{X}$ as certainly feasible, since its final $pu(\boldsymbol{X})$ value could become greater than or equal to 1.

Using the $pu(\boldsymbol{X})$ index, we now propose the following modification of the *RejectIndex* for constrained problems:

$$pup(\hat{f}, \boldsymbol{X}) = pu(\boldsymbol{X}) \cdot p(\hat{f}, \boldsymbol{X}),$$

where $\hat{f}$ is a parameter of this indicator, which is usually an approximation of f^*. This new index works like $p(\hat{f}, \boldsymbol{X})$ if $\boldsymbol{X}$ is certainly feasible, but if the box is undetermined, then it takes the feasibility degree of the box into account: the less feasible the box is, the less the value of $pup(\hat{f}, \boldsymbol{X})$ is.

Now we propose the following new interval selection criteria based on the $pup(\hat{f}, \boldsymbol{X})$ index:

C5: select the box with the highest $pup(\hat{f}, \boldsymbol{X})$ value, or
C6: a hybrid rule as C4, but using $pup(\hat{f}, \boldsymbol{X})$ instead of $pf^*(\boldsymbol{X})$.

Finally, to select a box which has both a low $\underline{\boldsymbol{f}}(\boldsymbol{X})$ value and a large $pup(\hat{f}, \boldsymbol{X})$ value, the new index:

$$pupb(\hat{f}, \boldsymbol{X}) = \begin{cases} \dfrac{\underline{\boldsymbol{f}}(\boldsymbol{X})}{pup(\hat{f}, \boldsymbol{X})} & \text{if } pup(\hat{f}, \boldsymbol{X}) \neq 0, \text{and} \\ M & \text{otherwise} \end{cases}$$

could be used, where M is a preset large value. Using this index, the following interval selection criteria arise:

C7: to select the box with the smallest $pupb(\hat{f}, \boldsymbol{X})$ value, and
C8: a hybrid rule similar to C4, but using the minimal $pupb(\hat{f}, \boldsymbol{X})$ values instead of the maximal $pf^*(\boldsymbol{X})$.

Note that $pup(\hat{f}, \boldsymbol{X})$ is 0 if $p(\hat{f}, \boldsymbol{X})$ or $pu(\boldsymbol{X})$ are 0. In either case, such a box does not seem to be suitable for being selected. For instance, if we consider $\hat{f}$ to be $\tilde{f}$, then $p(\tilde{f}, \boldsymbol{X}) = 0$ if and only if $\underline{\boldsymbol{f}}(\boldsymbol{X}) = \tilde{f}$, thus no point in $\boldsymbol{X}$ can improve the actual $\tilde{f}$; on the other hand, $pu(\boldsymbol{X}) = 0$ if and only if $\underline{\boldsymbol{g}}_j(\boldsymbol{X}) = 0$ for a constraint. That is why in those cases $pupb(\hat{f}, \boldsymbol{X})$ is set to a large M value.

We now consider the adaptive multisection for constrained problems. If a box $\boldsymbol{X}$ is feasible, the number of subboxes into which it must be subdivided can be given based on $p(\tilde{f}, \boldsymbol{X})$, but if the box is undetermined, it should be given by a modification of $p(\tilde{f}, \boldsymbol{X})$ which takes the feasibility degree of the box into account: the less feasible the box is, the more the modification on $p(\tilde{f}, \boldsymbol{X})$ should be made.

The new $pup(\tilde{f}, \boldsymbol{X})$ index satisfies both conditions. If $0 \leq P_1 < P_2$ are two given values considered as thresholds of the new adaptive multisection we propose to subdivide the current box

(a) using the traditional bisection by the widest interval if $pup(\tilde{f}, \boldsymbol{X}) < P_1$,
(b) halving the two widest interval components (resulting in 4 subboxes) if $P_1 \leq pup(\tilde{f}, \boldsymbol{X}) \leq P_2$, or
(c) dividing the two widest interval components into 3 parts (resulting in 9 subboxes) if $pup(\tilde{f}, \boldsymbol{X}) > P_2$.

In Section 2, we pointed out that $p(\tilde{f}, \boldsymbol{X})$ may be greater than 1 for some undetermined boxes. Concerning the selection criteria C3 to C8, this may provoke the selection of these boxes sooner than expected as compared with boxes for which $p(\tilde{f}, \boldsymbol{X}) < 1$. For the new adaptive multisection rules it may cause the subdivision of these boxes into an unnecessarily high number of subboxes as compared with the boxes for which $p(\tilde{f}, \boldsymbol{X}) < 1$ holds. To cope with such problems, the selection criterion and the kind of subdivision to be performed on that boxes could be determined by the feasibility index $pu(\boldsymbol{X})$ alone (i.e., considering $p(\tilde{f}, \boldsymbol{X}) := 1$). In order to reduce the number of undetermined boxes in $\mathcal{L}_{\mathcal{W}}$ with $p(\tilde{f}, \boldsymbol{X}) > 1$, it is advisable to update the $\tilde{f}$ value as quickly as possible.

4. Convergence analysis

4.1. Convergence analysis of the new subinterval selection rules

Convergence properties for branch–and–bound algorithms were presented by several earlier works. In particular, Horst and Tuy [25] investigated a general branch–and–bound frame for global optimization (involving problems in the form of (1)). They call a bounding operator (providing lower and upper bounds of the minimum of the objective function over the argument search set) to be *consistent*, if for each infinite and nested sequence of search subregions generated by the algorithm the sequence of differences between the current best known upper bound of f^* (i.e., $\tilde{f}$) and the computed lower bound of f over the particular subregions converges to 0. (The consistency is ensured if the same property holds for the difference *between the computed upper and lower bounds* of f over the particular subregions.) Moreover, a selection criterion is said to be *bound improving*, if each time after a finite number of iterations at least one search subregion providing the current best known lower bound of f^* is selected for subdivision. The consistency and the bound improving property together guarantee the convergence of the B&B procedure (in the sense that the updated lower and upper bounds of f^* are both converging to the global minimum value). For interval methods, it is easy to see that the zero-convergence of the inclusion function $\boldsymbol{f}$ of the objective function f implies consistency, and moreover, both selection criteria C1 and C2 are bound improving. (Note, that the above statement for the global convergence requires an additional simple property of the computed bounds, which corresponds to the inclusion isotonicity of $\boldsymbol{f}$.)

Similar classical results stated explicitly for interval B&B methods for criteria C1 and C2 are given e.g. by Ratschek and Rokne [37]. The convergence speed of interval algorithms with the above two rules were studied by Csallner and Csendes [8]. Methods using criterion C3 were investigated by Csendes [11].

In this section, results concerning interval algorithms using criteria C5 and C7 are presented. They are based on those obtained in [11] for criterion C3. Some of our results apply for algorithms involving *compound selection strategies*. For a compound strategy, we specify a *secondary selection rule* when more than one subbox has the best indicator value according to the first (or primary) selection criterion. Furthermore, we assume that if ties occur either for a single selection strategy, or in both criteria for a compound strategy, then any of these tied boxes can be selected for subdivision.

To investigate the convergence properties, we assume that the stopping conditions cannot be fulfilled. The algorithms will be studied without accelerating devices, with the exception of the feasibility test. The convergence of the algorithm variants are studied in the following sense:

Definition 5. *Let us consider a branch–and–bound algorithm variant for solving a given optimization problem in the form of (1). Let us denote A the* set of accumulation points *of the sequence of the leading boxes generated by the algorithm. Then we say that the particular algorithm variant* converges *to A for the given problem.*

When we use the terminology ‘the algorithm converges to a certain set of points’ (in general) we mean that for *each* global optimization problem (in the form of (1)) the algorithm converges to the specified set of points. In contrast to that, we say that ‘the algorithm may converge to a certain set of points’ if there exists a problem in the form of (1) for which the algorithm converges to the particular points.

Let f_k denote the updated approximation of f^* used in the k-th iteration for the $p(f_k, \boldsymbol{X})$ algorithm parameter. The following theorem investigates the constrained case when the sequence $\{f_k\}$ converges to a value $\hat{f}$ larger than the global minimum value.

Theorem 1. *Assume that the inclusion function of the objective function is isotone and it has the zero convergence property, and the inclusion functions of the constraint functions are either (a) isotone, or (b) α-convergent (zero convergent), but not necessarily isotone. Moreover, assume that the $p(f_k, \boldsymbol{Y})$ parameters are calculated with the f_k values converging to $\hat{f} > f^*$, for which there exists a feasible point $\hat{x} \in \boldsymbol{X}_0$ with $f(\hat{x}) = \hat{f}$. Then the interval branch-and-bound algorithm that selects that interval $\boldsymbol{Y}$ from the working list which has the maximal $pup(f_k, \boldsymbol{Z})$ value (i.e., which uses criterion C5) may converge to a feasible point*

$\hat{x} \in \boldsymbol{X}_0$ for which $f(\hat{x}) > f^*$, i.e., to a point which is not a global minimizer point of the problem.

Proof. Following the proof of Theorem 1 in [11], let the inclusion function of the objective function be defined as follows: for all $\boldsymbol{X} \in \boldsymbol{X}_0$ let $d(\boldsymbol{X}) = \max(\overline{f}(\boldsymbol{X}) - \hat{f}, \hat{f} - \underline{f}(\boldsymbol{X}))$. For the starting box let $\boldsymbol{f}(\boldsymbol{X}_0) = [\hat{f} - 9d(\boldsymbol{X}_0), \hat{f} + d(\boldsymbol{X}_0)]$. For those $\boldsymbol{X}_i$ subboxes which contain $\hat{x}$, the inclusion function is given by $\boldsymbol{f}(\boldsymbol{X}_i) = [\hat{f} - 9d(\boldsymbol{X}_i), \hat{f} + d(\boldsymbol{X}_i)]$. For all other subintervals $\boldsymbol{Y}$ let $\boldsymbol{f}(\boldsymbol{Y}) = [\hat{f} - d(\boldsymbol{Y}), \hat{f} + d(\boldsymbol{Y})]$. The defined function is an interval inclusion function and it has the isotonicity property, and moreover, it is zero convergent for all interval sequences $\boldsymbol{X}_k$ with $\boldsymbol{f}(\boldsymbol{X}_k) \to \hat{f}$. Additionally, since f_k converges to $\hat{f}$, after some iteration steps the $p(f_k, \boldsymbol{X})$ value on the box having $\hat{x} \in \boldsymbol{X}$ will be close enough to 0.9 to prefer that box for selection against the other boxes not containing $\hat{x}$, which have $p(f_k, \boldsymbol{X})$ values close to 0.5. From that time the subbox containing $\hat{x}$ would be selected for further subdivision by the rule C3.

For constrained problems and for the selection criterion C5 we can easily extend the the mentioned example under the different conditions (a) and (b):

(a) Let the inclusion functions of the constraints be generated by natural interval extension, this extension is known to be inclusion isotone. Let each point in $\boldsymbol{X}_0$ be strictly feasible, and let $pu(\boldsymbol{X}_0) = 1$. Then $pup(f_k, \boldsymbol{Z}) = p(f_k, \boldsymbol{Z})$ holds for all $\boldsymbol{Z} \subseteq \boldsymbol{X}_0$ subboxes since the inclusion isotonicity of the constraint functions implies $pu(\boldsymbol{Z}) = 1$. Thus, the sequence of the selected subboxes will be the same as it is in the unconstrained case. This proves the theorem for condition (a).

(b) For simplicity, consider a problem having only one constraint, $g(x) \leq 0$. The construction of a suitable inclusion function for the constraint function is as follows: first, apply the natural interval extension to produce an interval inclusion function denoted by $\boldsymbol{g}$. This function is known to be isotone and α-convergent with $\alpha = 1$: $w(\boldsymbol{g}(\boldsymbol{X})) - w(g(\boldsymbol{X})) \leq Cw(\boldsymbol{X})$ for a suitable constant C. Now modify $\boldsymbol{g}$ for each $\boldsymbol{X} \subseteq \boldsymbol{X}_0$ to get an other interval inclusion function (denoted by $\boldsymbol{k}$) in the following way:

(1) If $\underline{\boldsymbol{g}}(\boldsymbol{X}) > 0$, then let $\boldsymbol{k}(\boldsymbol{X}) = \boldsymbol{g}(\boldsymbol{X})$. Subintervals like this will be eliminated by the feasibility test.
(2) If $\underline{\boldsymbol{g}}(\boldsymbol{X}) \leq 0$ and $\overline{\boldsymbol{g}}(\boldsymbol{X}) > 0$, then denote the maximal absolute value of the interval $\boldsymbol{g}(\boldsymbol{X})$ by $d(\boldsymbol{X})$, and let $\boldsymbol{k}(\boldsymbol{X}) = [-9d(\boldsymbol{X}), d(\boldsymbol{X})]$ if $\hat{x} \in \boldsymbol{X}$ (resulting in $pu(\boldsymbol{X}) = 0.9$ for this kind of subboxes), and let $\boldsymbol{k}(\boldsymbol{X}) = [-d(\boldsymbol{X}), d(\boldsymbol{X})]$ if $\boldsymbol{X}$ does not contain $\hat{x}$ (resulting in $pu(\boldsymbol{X}) = 0.5$ in the latter case).
(3) If $\overline{\boldsymbol{g}}(\boldsymbol{X}) \leq 0$, set $\boldsymbol{k}(\boldsymbol{X}) = \boldsymbol{g}(\boldsymbol{X})$. This results in $pu(\boldsymbol{X}) = 1$.

Obviously, $\boldsymbol{k}$ will be an inclusion function of the constraint function. The α-convergence of $\boldsymbol{k}$ (with $\alpha = 1$) can be proved by the following: for simplicity, assume that the constraint function is a rational function; then for its natural interval extension $w(\boldsymbol{g}(\boldsymbol{X})) \leq Mw(\boldsymbol{X})$ holds for all $\boldsymbol{X} \subseteq \boldsymbol{X}_0$ with a positive constant M (this is a kind of a Lipschitz condition for interval functions, for details, see e.g. [36]).

Since for each subbox $w(\boldsymbol{k}(\boldsymbol{X})) \leq 10w(\boldsymbol{g}(\boldsymbol{X}))$ holds from the definition of $\boldsymbol{k}$, by the application of the α-convergence of $\boldsymbol{g}$ we obtain:

$$w(\boldsymbol{k}(\boldsymbol{X})) - w(g(\boldsymbol{X})) \leq w(\boldsymbol{g}(\boldsymbol{X})) - w(g(\boldsymbol{X})) + 9w(\boldsymbol{g}(\boldsymbol{X})) \leq$$

$$\leq Cw(\boldsymbol{X}) + 9w(\boldsymbol{g}(\boldsymbol{X})) \leq (C + 9M)w(\boldsymbol{X}),$$

Thus, $\boldsymbol{k}$ is α-convergent with $\alpha = 1$. (Note, that by definition the α-convergence implies the zero convergence property.)

We can demonstrate that the isotonicity of the inclusion function of the objective is not required in this part of the proof. Namely, the resulting function $\boldsymbol{k}$ can be changed on a finite number of subboxes to obtain a non-isotone inclusion function having α-convergence. For example we may multiply the bounds of $\boldsymbol{k}$ on some subboxes in case (2) by a suitable number, then the pu values of these subboxes will be unchanged, but the isotonicity will fail. This has obviously no effect on the α-convergence property of $\boldsymbol{k}$.

Now apply the constructed $\boldsymbol{k}$ by following the line of thought of the unconstrained case: after some iteration steps the $pup(f_k, \boldsymbol{X}) = p(f_k, \boldsymbol{X}) \cdot pu(\boldsymbol{X})$ value on the box having $\hat{x} \in \boldsymbol{X}$ will be close enough either to $0.9 \cdot 0.9 = 0.81$ or to $0.9 \cdot 1 = 0.9$ to prefer that box for selection against the other boxes having $pup(f_k, \boldsymbol{X})$ values close either to $0.5 \cdot 0.5 = 0.25$ or to $0.5 \cdot 1 = 0.5$. From that iteration only the subboxes containing $\hat{x}$ will be selected for further subdivision, which proves the convergence of the algorithm to $\hat{x}$ under condition (b). $\square$

Remark 1. The cut-off test and the zero convergence property of $\boldsymbol{f}$ *can be* sufficient to ensure convergence to global minimizer points even if the sequence $\{f_k\}$ does not converge to f^*: for all interval subsequences of the actual intervals, if they converge to a point for which the objective value is greater than f^*, then the zero convergence property will imply $\underline{\boldsymbol{f}}(\boldsymbol{X}_k) > f^*$, i.e., in case $\tilde{f}$ approximates f^* in such a way that also $\underline{\boldsymbol{f}}(\boldsymbol{X}_k) > \tilde{f}$ holds, then such boxes will be deleted.

The following lemma contains a characterization of a special kind of sequence of nested subboxes, which should be discussed separately from other cases while studying convergence properties.

Lemma 1. *Assume that the interval inclusion functions of the constraint functions are inclusion isotone. Let $\{\boldsymbol{X}_k\}$ be a nested subbox sequence generated by the branch-and-bound algorithm that converges to a feasible point x. Then the following three statements are equivalent:*

1. *For a suitable number i, $pu(\boldsymbol{X}_i) = 0$.*
2. *For a suitable number i, $pu(\boldsymbol{X}_k) = 0$ $\forall k \geq i$.*
3. *For a suitable i and $\forall k \geq i$ the subbox $\boldsymbol{X}_k$ does not contain strictly feasible points (for which $g_j(x) < 0$ would hold), and moreover, there exists a constraint g_s having $g_s(x) = 0$ and providing interval inclusions without overestimation on the lower bound over all these subboxes.*

Proof. $1 \Rightarrow 3$: If statement 1 holds then $\underline{\boldsymbol{g}}_s(\boldsymbol{X}_i) = 0$ must hold for an index s. Since all the boxes from $\{\boldsymbol{X}_k\}$ contain at least one feasible point (e.g. x), and $\boldsymbol{g}_s$ is inclusion isotone, $\underline{\boldsymbol{g}}_s(\boldsymbol{X}_k) = 0$ and thus $\boldsymbol{g}_s(\boldsymbol{X}_k) \geq 0$ will hold for all $k \geq i$. This means that from the ith iteration, the subboxes cannot contain strictly feasible points. On the other hand, on subboxes containing a feasible point but not having strictly feasible points, $\underline{g}_s(\boldsymbol{X}_k) = 0$ holds where $g_s(\boldsymbol{X}_k)$ is the range of g_s over $\boldsymbol{X}_k$. Furthermore, from the inclusion properties $g_s(x) = 0$ should hold for that constraint.

$3 \Rightarrow 2$: Assume now that statement 3 holds. If $\boldsymbol{g}_s$ has no overestimation on subboxes not containing strictly feasible points but containing x, then $\underline{\boldsymbol{g}}_s(\boldsymbol{X}_k) = 0$, and thus $pu(\boldsymbol{X}_k) = 0$, $\forall k \geq i$.

$2 \Rightarrow 1$: This implication obviously holds. □

We introduce a notation used in the theorems below: we investigate nested subinterval sequences (converging to a point in $\boldsymbol{X}_0$) as a subsequence of the leading boxes generated by the algorithm, i.e. as a subsequence of the sequence of boxes which are selected by the algorithm for subdivision. For such a subsequence (denoted for example by $\{\boldsymbol{X}_l\}$, $l = 1, 2, \ldots$) we can assume that its elements are selected at iteration steps $k_1, k_2, \ldots$, thus, the corresponding $p(f_k, \boldsymbol{Y})$ and $pup(f_k, \boldsymbol{Y})$ values of these subboxes can be denoted by $p(f_{k_l}, \boldsymbol{X}_l)$ and $pup(f_{k_l}, \boldsymbol{X}_l)$ for $l = 1, 2, \ldots$, respectively.

Theorem 2. *Assume that the inclusion functions of the objective function and the constraint functions have the zero convergence property, no interval sequence* $\{\boldsymbol{X}_k\}_{k=1}^{\infty}$, $\boldsymbol{X}_k \subseteq \boldsymbol{X}_0$ *contains a subsequence for which* $\lim_{l\to\infty} pu(\boldsymbol{X}_{k_l}) = 0$ *and that* f_k *converges to* $\hat{f} < f^*$. *Then the interval branch-and-bound algorithm that selects that box* $\boldsymbol{Y}$ *from the working list which has the maximal* $pup(f_k, \boldsymbol{Z})$ *value (i.e., which uses criterion C5) converges to each feasible point of the search region* $\boldsymbol{X}_0$ *regardless of the objective function value.*

Proof. As it was shown for the unconstrained case in the proof of Theorem 2 in [11], for any interval subsequence $\{\boldsymbol{X}_l\}$ converging to a point $x \in \boldsymbol{X}_0$, the corresponding sequence $\{p(f_{k_l}, \boldsymbol{X}_l)\}$ converges to minus infinity.

Now we investigate the behavior of the $pup(f_{k_l}, \boldsymbol{X}_l)$ values. We state that $\lim_{l\to\infty} pup(f_{k_l}, \boldsymbol{X}_l) = -\infty$. To prove this, we consider the following two cases:

- If $\{\boldsymbol{X}_l\}$ converges to a strictly feasible point x, then, after a finite number of iterations, $\boldsymbol{g}_j(\boldsymbol{X}_l) < 0$ holds for each j, $j = 1, \ldots, r$ (due to the zero convergence property of the inclusion function $\boldsymbol{g}_j$), which implies $pu(\boldsymbol{X}_l) = 1$. Hence, $\lim_{l\to\infty} pup(f_{k_l}, \boldsymbol{X}_l) = \lim_{l\to\infty} p(f_{k_l}, \boldsymbol{X}_l) = -\infty$.
- Now suppose that $\{\boldsymbol{X}_l\}$ converges to a feasible point x for which $g_j(x) = 0$ for some index j. Note that each $\{\boldsymbol{X}_l\}$ subsequence of leading boxes can only converge to feasible points: otherwise, due to the zero convergence property of the constraint functions, after a certain number of iterations $\boldsymbol{g}_j(\boldsymbol{X}_l) > 0$ will hold for a constraint indexed by j; at that point $\boldsymbol{X}_l$ will be removed by the feasibility test. Now since no interval sequence $\{\boldsymbol{X}_k\}$ contains a subsequence for which $\lim_{l\to\infty} pu(\boldsymbol{X}_{k_l}) = 0$, $\lim_{l\to\infty} pup(f_{k_l}, \boldsymbol{X}_l) = -\infty$.

In this way no box $\boldsymbol{Y}$ having positive width will remain in $\mathcal{L}_{\mathcal{W}}$, since after a finite number of iterations its $pup(f_k, \boldsymbol{Y})$ value will be the largest, and hence, the box $\boldsymbol{Y}$ will be appointed for subdivision. This proves the statement of the theorem. $\square$

The condition that no interval sequence $\{\boldsymbol{X}_k\}$ contains a subsequence for which $\lim_{k\to\infty} pu(\boldsymbol{X}_{k_l}) = 0$ seems to be a strong one. Still it is easy to see that for all problems not satisfying it the related inclusion function can be changed in a suitable way to fulfill this criterion. Consider that the condition is hurt for the constraint function $g_j(x)$ and $\boldsymbol{g}_j(\boldsymbol{X}) \not\equiv [0,0]$. Then redefine the related inclusion function as $\boldsymbol{g}'_j(\boldsymbol{X}) = [-a, a]$ where $a = \max\{|\underline{\boldsymbol{g}}_j(\boldsymbol{X})|, |\overline{\boldsymbol{g}}_j(\boldsymbol{X})|\}$. Obviously $\boldsymbol{g}_j(\boldsymbol{X}) \subseteq \boldsymbol{g}'_j(\boldsymbol{X})$, $pu_{\boldsymbol{g}'_j}(\boldsymbol{X})) = 1/2$, and $\boldsymbol{g}'_j$ keeps the isotonicity, zero convergence and α-convergence properties of $\boldsymbol{g}$ (the last one with a larger K value). Also the $\boldsymbol{g}_j(\boldsymbol{X}) \equiv [0,0]$ case can be handled with a proper overestimation.

The consequence of the previous theorem is that if the $\{f_k\}$ sequence is set (either by the user or by the algorithm itself) to converge to a value lower than the optimal value, then the algorithm will converge either to all feasible points of the search region or only to a certain number of feasible points for which at least one of the constraints is active. In the first case the $pup(f_{k_l}, \boldsymbol{X}_l)$ values will converge to minus infinity for all the converging subbox subsequences. In the latter case there exists at least one subsequence having a finite limit of its $pup(f_{k_l}, \boldsymbol{X}_l)$ values.

Remark 2. From the previous theorem one can build problems in which the convergence either to all feasible points of the search region, or only to some feasible points for which at least one of the constraints is active depends on the type of the subdivision rule, (i.e. how to split a given box) and on the search area. Thus, in some cases a differing type of splitting method or a suitable search area can basically change the convergence properties of the algorithm when $f_k \to \hat{f} < f^*$.

In the next theorem we give necessary and sufficient conditions for the global convergence of accelerated algorithms. The conditions are based on a suitable choice of the f_k values.

Theorem 3. *Assume that the inclusion functions of both the objective function and the constraint functions are isotone and have the zero convergence property. Consider the interval branch-and-bound algorithm that uses the feasibility test (and optionally other discarding tests) and that selects as next actual box that subinterval $\boldsymbol{Y}$ from the working list which has the maximal $pup(f_k, \boldsymbol{Z})$ value as first criterion, and the box which has the maximal $p(f_k, \boldsymbol{Z})$ value as second criterion. Then a necessary and sufficient condition for the convergence of this algorithm to a set of global minimizer points is that the sequence $\{f_k\}$ converges to the global minimum value f^* and there exist at most a finite number of f_k values below f^*.*

Proof. 1. Consider first the sufficiency of the conditions. Assume that although the conditions for the sequence $\{f_k\}$ are fulfilled, still the sequence of leading boxes generated by the algorithm has a subsequence $\{\boldsymbol{X}_l\}$ converging to a point x' which is not optimal, i.e. $f(x') > f^*$. Notice that if x' is not feasible then for a suitable value of l the assertion $\boldsymbol{g}_j(\boldsymbol{X}_l) > 0$ will hold for some j due to the zero convergence property of the constraint functions, and thus, $\boldsymbol{X}_l$ would be eliminated by the feasibility test obstructing the convergence to x'. Thus, we can assume that x' is feasible.

First, assume that the case of Lemma 1 does not hold for $\{\boldsymbol{X}_l\}$, i.e. $pu(\boldsymbol{X}_l) > 0$ stands for all l. Then $\underline{\boldsymbol{f}}(\boldsymbol{X}_l) \to f(x')$ as a consequence of the zero convergence property of $\boldsymbol{f}$. Since $f_k \to f^*$, there exists a positive integer i such that $\underline{\boldsymbol{f}}(\boldsymbol{X}_l) > f_{k_l}$, when $l > i$. This means that from some iteration i' the element of $\{\boldsymbol{X}_l\}$ which is currently in the working list has negative $p(f_{k_l}, \boldsymbol{X}_l)$ value. Together with $pu(\boldsymbol{X}_l) > 0$ we obtain $pup(f_{k_l}, \boldsymbol{X}_l) < 0$ after iteration i'. (Note, that $i = i'$ does not necessarily hold. The value i' gives an iteration number of the algorithm, while i means the serial number when an $\boldsymbol{X}_l$ is selected for subdivision.)

On the other hand, one can easily see that a subinterval containing an optimal solution x^* must always exist in the working list. This assertion is based on the fact that a subinterval $\boldsymbol{Y}$ containing x^* cannot be eliminated by any of the discarding tests. Investigating the $pup(f_k, \boldsymbol{Y})$ values for subintervals $\boldsymbol{Y}$ containing x^*, we can state $f^* - \underline{\boldsymbol{f}}(\boldsymbol{Y}) \geq 0$ because of the inclusion property of $\boldsymbol{f}$. This implies $f_k - \underline{\boldsymbol{f}}(\boldsymbol{Y}) \geq 0$ when $f_k \geq f^*$, that is, $p(f_k, \boldsymbol{Y})$ will be nonnegative after a finite number i^* of iteration steps. Additionally, $pu(\boldsymbol{Y}) \geq 0$ and then $pup(f_k, \boldsymbol{Y}) \geq 0$ holds for all boxes of this type.

Consequently, from the iteration step $\max(i', i^*)$ at each iteration the current box $\boldsymbol{X}_l$ (containing x' and having a negative *pup* value) will be compared to the other subintervals of $\mathcal{L}_{\mathcal{W}}$ including a subinterval containing x^* that has a nonnegative *pup* value. Then due to the first selection criterion the algorithm will not choose $\boldsymbol{X}_l$. But this contradicts our original assumption about the convergence of the algorithm to a non-minimizer point (through a subinterval sequence $\{\boldsymbol{X}_l\}$ not fitting the statements of Lemma 1).

Let us now consider a subsequence $\{\boldsymbol{X}_l\}$ of the leading boxes converging to a non-optimal feasible point x' and satisfying one of the statements of Lemma 1. Then after a finite number i' of iterations, at each iteration step $pu(\boldsymbol{X}_l) = pup(f_{k_l}, \boldsymbol{X}_l) = 0$ and $p(f_{k_l}, \boldsymbol{X}_l) < 0$ hold for the actual element $\boldsymbol{X}_l$ of the sequence.

Nevertheless, also in this case a subinterval containing x^* must exist in the working list having a nonnegative *pup* value when the iteration counter is greater than i^*. After $\max(i', i^*)$ iterations a box $\boldsymbol{X}_l$ containing x' and having $pup(f_{k_l}, \boldsymbol{X}_l) = 0$ will be compared also to a subinterval $\boldsymbol{Y}$ containing x^* and having a nonnegative *pup* value. But also in this case $\boldsymbol{X}_l$ cannot be preferred by the interval selection step: if $pup(f_{k_l}, \boldsymbol{Y}) > 0$, then the first selection criterion finds $\boldsymbol{Y}$ better than $\boldsymbol{X}_l$. If $pup(f_{k_l}, \boldsymbol{Y}) = pup(f_{k_l}, \boldsymbol{X}_l) = 0$, then the second criterion finds $\boldsymbol{Y}$ to be a better choice than $\boldsymbol{X}_l$, since $\boldsymbol{Y}$ has a nonnegative $p(f_{k_l}, \boldsymbol{Y})$ value, while $\boldsymbol{X}_l$ has a negative $p(f_{k_l}, \boldsymbol{X}_l)$ value. We have proven a contradiction also for the case when the sequence $\{\boldsymbol{X}_l\}$ satisfies the statements of Lemma 1. Thus, if $\{f_k\}$ converges to the global minimum value and there exist at most a finite number of f_k values below f^*, then the accelerated algorithm will converge exclusively to a set of global minimizer points.

2. The necessity of the first condition (i.e., that the sequence $\{f_i\}$ must converge to the global minimum value f^*) follows from Theorems 1 and 2: Theorem 1 show examples for which the algorithm converges to a nonoptimal point when $f_i \to \hat{f} > f^*$ holds, consequently, if our algorithm converges to a set of global optimizer points for each optimization problem investigated, then necessarily $f_i \to \hat{f} \leq f^*$ must hold. In a similar way, from Theorem 2 $f_i \to \hat{f} \geq f^*$ follows in the sense that if $f_i \to \hat{f} < f^*$, then there exist problems for which the algorithm does not converge to a set of global optimizer points.

Next we prove by means of a counter example that to have the convergence of the algorithm to a set of global minimizer points, it is necessary that at most a finite number of f_k values is below f^*.

Consider a problem for which there exists a strictly feasible point $x' \in \boldsymbol{X}_0$ with $f(x') > f^*$ and a strictly feasible box $\boldsymbol{Y} \subset \boldsymbol{X}_0$ for which $f(y) = f^*$ holds for each $y \in \boldsymbol{Y}$. It is obvious that the $pup(f_i, \boldsymbol{Y}'_i)$ values will converge to $-\infty$ as the subintervals $\boldsymbol{Y}'_i$ converge to x' (notice that after some iteration $pu(\boldsymbol{Y}'_i) > 0$ and $p(f_i, \boldsymbol{Y}'_i)$ converges to $-\infty$, since the numerator of $p(f_i, \boldsymbol{Y}'_i)$ will remain negative after some iteration and the denominator converges to zero). Having a fixed sequence of $\{f_i\}$ values converging to f^*, and with an infinite number of f_i below f^*, the lower bounds $\underline{\boldsymbol{f}}(\boldsymbol{Y}_i)$ of the subboxes $\boldsymbol{Y}_i$ within $\boldsymbol{Y}$ can be set (similarly to the construction of the inclusion functions in the proof of Theorem 1) in such a way that the resulting $pup(f_i, \boldsymbol{Y}_i)$ value will be less than the $pup(f_i, \boldsymbol{Y}'_i)$ value for the subinterval containing the x' point. Furthermore, this can be done even in accordance with the required inclusion isotonicity and zero convergence properties of the related inclusion function (notice that $pu(\boldsymbol{Y}_i) > 0$ will hold after some iteration, due to the strict feasibility of $\boldsymbol{Y}$ and the zero convergence property of the inclusion functions of the constraint functions, and we can set the lower bounds $\underline{\boldsymbol{f}}(\boldsymbol{Y}_i)$ greater than f_i but less that f^* since we are assuming that $f_i < f^*$ for an infinite number of cases). Thus the algorithm converges also to x', which is not a global minimizer point. □

Under the conditions of the previous theorem, the set of accumulation points A of the leading boxes of the algorithm is a subset of the set of global minimizer points X^* of the given problem. The equality $A = X^*$ does not hold in general due to the 'hidden' global minimizer points [42].

Applying the above theorem for the most commonly used way of computing an approximation of f^*, we can state the following corollary:

Corollary 1. *If our algorithm uses the feasibility test (and optionally other discarding tests) as accelerating devices, and selects as next leading box that box $\boldsymbol{Y}$ from the working list which has the maximal $pup(\tilde{f}, \boldsymbol{Z})$ value as first criterion, and the box which has the maximal $p(\tilde{f}, \boldsymbol{Z})$ value as second criterion, where $\tilde{f}$ is the best upper bound for the global minimum value, and* its convergence to f^* can be ensured*, then the algorithm converges exclusively to global minimizer points.*

Since the convergence of $\tilde{f}$ to f^* is not always ensured, also other approaches have to be studied. Dealing with a compound selection criterion, the next theorem characterizes the convergence properties of the algorithms which use a selection criterion based on C1 and C5 through a utility function $u : \mathbb{R}^2 \to \mathbb{R}$, which selects the box for which $u(pup(f_k, \boldsymbol{Z}), -\underline{\boldsymbol{f}}(\boldsymbol{Z}))$ is maximal.

Theorem 4. *Assume that the inclusion functions of both the objective function and the constraint functions are isotone and have the zero convergence property. Consider an interval branch-and-bound algorithm that uses the feasibility test (and optionally other discarding tests) and which selects as next actual box that box $\boldsymbol{Y}$ from the working list which has the maximal $u(pup(f_k, \boldsymbol{Z}), -\underline{\boldsymbol{f}}(\boldsymbol{Z}))$ value as first criterion, and the box which has the maximal $p(f_k, \boldsymbol{Z})$ value as second criterion.*

(a) Then the sufficient conditions for the convergence of the algorithm to a set of global minimizer points are that the sequence $\{f_k\}$ converges to the global minimum value f^*, there exist at most a finite number of f_k values below f^* and that the utility function $u(x, y)$ is strictly monotonically increasing in both of its arguments.

(b) On the other hand, $f_i > f^* + \delta$ for a $\delta > 0$ allows convergence to a non-optimal point even for the class of the above defined utility functions satisfying the condition stated in (a).

Proof. (a) Assume that $f_k \to f^*$, and there exists at most a finite number of f_k values below f^*, but still there is a subsequence $\{\boldsymbol{X}_l\}$ of the leading boxes which converges to a non-optimal point x'. Again, x' can be considered as a feasible point.

In the working list there is always a subinterval $\boldsymbol{Y}$ containing an optimizer x^* since such cannot be deleted by the discarding tests. For this type of subintervals $pup(f_k, \boldsymbol{Y}) \geq 0$ holds after a suitable number of iterations (see the proof of Theorem 3).

On the other hand, after a suitable number i' of iteration steps $pup(f_{k_l}, \boldsymbol{X}_l) < 0$ for the subsequence $\{X_l\}$, i.e. a non-positive pup value will occur. Additionally, after a sufficiently large number of iterations for the actual $\boldsymbol{Y}$ containing x^* and for the actual element of $\{\boldsymbol{X}_l\}$, the inequality $\underline{\boldsymbol{f}}(\boldsymbol{Y}) < \underline{\boldsymbol{f}}(\boldsymbol{X}_l)$ holds, which implies

$$u(pup(f_{k_l}, \boldsymbol{Y}), -\underline{\boldsymbol{f}}(\boldsymbol{Y})) > u(pup(f_{k_l}, \boldsymbol{X}_l), -\underline{\boldsymbol{f}}(\boldsymbol{X}_l)) \tag{3}$$

because of the strict monotonicity of u. Note, that if both $pup(f_{k_l}, \boldsymbol{X}_l)$ and $pup(f_{k_l}, \boldsymbol{Y})$ are equal to 0, then the second argument of u guarantees the strict inequality in (3). Consequently $\boldsymbol{X}_l$ cannot be preferred against $\boldsymbol{Y}$ by the algorithm. This contradiction proves statement (a).

(b) In Theorem 4 of [11] the same statement was proved for the unconstrained case and for utility functions with the form of $u(p(f_k, \boldsymbol{X}), -\underline{\boldsymbol{f}}(\boldsymbol{X}))$, i.e. for a compound subinterval selection criterion which applies $p(f_k, \boldsymbol{X})$ instead of $pup(f_k, \boldsymbol{X})$ in the first argument.

Merely notice that we can simply extend this result to the constrained case and to utility functions $u(pup(f_k, \boldsymbol{X}), -\underline{\boldsymbol{f}}(\boldsymbol{X}))$. We use the same trick as we did in the proof of Theorem 1 of this paper. In details, create a constrained problem with constraints which leave each point to be feasible in $\boldsymbol{X}_0$ in such a way that $pu(\boldsymbol{X}_0) = 1$. This will imply $p(f_k, \boldsymbol{X}) = pup(f_k, \boldsymbol{X})$ and therefore $u(p(f_k, \boldsymbol{X}), -\underline{\boldsymbol{f}}(\boldsymbol{X})) = u(pup(f_k, \boldsymbol{X}), -\underline{\boldsymbol{f}}(\boldsymbol{X}))\ \forall \boldsymbol{X} \subseteq \boldsymbol{X}_0$. In other words, the algorithm generates the same subinterval sequence (converging to x') as in the unconstrained case. □

Let us consider now the subinterval selection based on criterion C7, i.e. the method which selects that subinterval $\boldsymbol{X}$ from the working list for which the $pupb(f_k, \boldsymbol{X})$ value is minimal. From the definition of $pupb(f_k, \boldsymbol{X})$, we can see that the restrictions for the sequence $\{f_k\}$ applied in Theorem 3 will not be sufficient conditions for the global convergence in this case. Namely, let us follow the first part of the proof of Theorem 3, and assume the existence of a subsequence $\{\boldsymbol{X}_l\}$ converging to a non-optimal x'. Then assuming $\underline{\boldsymbol{f}}(\boldsymbol{X}) > 0$ for all subintervals $\boldsymbol{X} \subseteq \boldsymbol{X}_0$, we obtain a negative $pupb(f_{k_l}, \boldsymbol{X}_l)$ value after some iterations because of the negative $pup(f_{k_l}, \boldsymbol{X}_l)$ values. Moreover, a subinterval $\boldsymbol{Y}$ containing a global optimizer will have positive $pup(f_k, \boldsymbol{Y})$ and then a positive $pupb(f_k, \boldsymbol{Y})$ value, thus after some iterations the subinterval containing x^* will not be preferred against a subinterval $\boldsymbol{X}_l$. As we see, the difference between the signs of the $pup(f_{k_l}, \boldsymbol{X}_l)$ and $pup(f_k, \boldsymbol{Y})$ values (which was the key to the proof of Theorem 3) may turn into a disadvantage in this case. Additionally, with a simple transformation on the objective function, e.g. by adding or subtracting a suitable constant, one can change the sign of $\underline{\boldsymbol{f}}$ on some subintervals. Since the transformation causes no changes in the $pup(f_k, \boldsymbol{Z})$ values (if $\{f_k\}$ is shifted accordingly), also the subinterval selection decisions based on $pupb(f_k, \boldsymbol{Z})$ can be altered.

Due to the same fact we cannot apply the previous theorem for selection criterion C7: Although $-pupb(f_k, \boldsymbol{X})$ has the form of a utility function $u(pup(f_k, \boldsymbol{X}), -\underline{\boldsymbol{f}}(\boldsymbol{X}))$, $-pupb(f_k, \boldsymbol{X})$ is not monotonically increasing in both of its argument in general: if $\underline{\boldsymbol{f}}(\boldsymbol{X})$ is fixed to be negative then for positive $pup(f_k, \boldsymbol{X})$ values the utility function

$$u(pup(f_k, \boldsymbol{X}), -\underline{\boldsymbol{f}}(\boldsymbol{X})) = -pupb(f_k, \boldsymbol{X}) = \frac{-\underline{\boldsymbol{f}}(\boldsymbol{X})}{pup(f_k, \boldsymbol{X})}$$

will be strictly monotonically decreasing in the first argument.

Nevertheless, we can give general sufficient conditions for the convergence of the algorithm when we additionally assume that $\{f_k\}$ is used by the cut-off test:

Theorem 5. *Assume that the inclusion functions of both the objective function and the constraint functions have the zero convergence property. Consider the interval branch-and-bound algorithm that uses the feasibility and the cut-off tests (and optionally other discarding tests) and that uses an arbitrary selection criterion (e.g. one of the rules C1–C8). Then a sufficient condition for the convergence of this algorithm to a set of global minimizer points is that the sequence* $\{f_k\}$ *converges to the global minimum value* f^*, *with the exception of a finite number of iteration steps at each iteration* $f_k = f(x_k)$ *holds for an* $x_k \in \boldsymbol{X}_0$ *feasible point, and these* $f(x_k)$ *values are used to update the* $\tilde{f}$ *cut-off value.*

Proof. The last assumption of the theorem implicitly says that after a certain number of iterations the currently used cut-off value of the algorithm can be set to a number less than or equal to f_k, i.e. it is possible to eliminate a box $\boldsymbol{Y}$ having $\underline{\boldsymbol{f}}(\boldsymbol{Y}) > f_k$ by the cut-off test.

Assume that although $f_k \to f^*$ and f_k can be used from iteration i as a cut-off value, still the algorithm generates a subsequence $\{\boldsymbol{X}_l\}$ of the leading boxes which converges to a non-optimal x' with the objective function value $f(x') = f' > f^*$. Again, x' should be considered as a feasible point (see the proof of Theorem 3). Since the objective function has the zero convergence property, $\underline{\boldsymbol{f}}(\boldsymbol{X}_l) \to f'$. Then after a finite number i' of iteration steps, at each iteration $\underline{\boldsymbol{f}}(\boldsymbol{X}_l) > f_{k_l}$ will hold for the current $k_l > i'$ values. Clearly, at latest at iteration step $\max(i, i')$ the current $\boldsymbol{X}_l$ will be deleted by the cut-off test, which proves the theorem by a contradiction. □

If the $\{f_k\}$ sequence of the previous theorem is associated with the best updated upper bound for the global minimum value, then the following corollary can be introduced:

Corollary 2. *Assume that our algorithm uses the feasibility and cut-off tests (and perhaps other discarding tests) as accelerating devices, and an arbitrary selection criterion with* $f_k := \tilde{f}$*, where* $\tilde{f}$ *is the best upper bound for the global minimum value. If* $\tilde{f} \to f^*$ can be ensured*, then the algorithm converges exclusively to global minimizer points.*

Remark 3. Notice, that in Theorem 5 we used in fact an additional restriction to $\{f_k\}$ as compared to Theorem 3. The existence of the above sequence $\{x_k\}$ providing $f(x_k) = f_k$ implies that $f_k \geq f^*$ with the exception of a finite number of iterations. Thus, there exist at most a finite number of f_k values below f^*. Additionally, in the latter theorem these f_k values are assumed to be used also as cut-off values.

Remark 4. One can see also from another point of view that the subinterval selection criterion C7 alone is not enough to provide convergence to x^*. The reason for that is that one can construct a global optimization problem in which $pup(f_i, \boldsymbol{Y}) = 0$ $(x^* \in \boldsymbol{Y})$, and then according to Lemma 1 $pupb(f_k, \boldsymbol{Y}) = M$ will hold when $k > i$ for all subintervals containing x^*. Thus, after a finite number of iterations such $\boldsymbol{Y}$ boxes will be the least preferred among all the subboxes in $\mathcal{L}_{\mathcal{W}}$. (A subsequence $\{\boldsymbol{Y}_k\}$ of the leading subboxes having this property will occur when the problem fulfills one of the assertions of Lemma 1.) To provide the convergence exclusively to global optimizers also in these cases (e.g. by eliminating subsequences converging to non-optimal points) we need the accelerating devices, especially the cut-off test.

As we have seen, the availability of a sequence converging to the global minimum is indispensable. One of the common estimators of the global minimum is the updated upper bound of it, defined by

$$\tilde{f}_k := \min\{\tilde{f}_{k-1}, \overline{\boldsymbol{f}}(\boldsymbol{Y}_1), \ldots, \overline{\boldsymbol{f}}(\boldsymbol{Y}_s)\}, \tag{4}$$

where $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_s$ are the result intervals after subdividing the leading interval $\boldsymbol{Y}$ in the iteration cycle k, and $\tilde{f}_0 := \overline{\boldsymbol{f}}(\boldsymbol{X}_0)$.

Another estimator of the global minimum is the sequence of the lower bounds computed on the leading intervals, i.e.

$$\underline{\boldsymbol{f}}(\boldsymbol{X}_k), \tag{5}$$

where $\boldsymbol{X}_k$ is the leading interval in the iteration cycle k. When using the selection criterion C1, the sequence $\{\tilde{f}_k\}$ as described in (4) converges to f^* from above and the sequence $\{\underline{\boldsymbol{f}}(\boldsymbol{X}_k)\}$ of (5) converges to f^* from below (see [37]). These assertions do not hold any more when using the selection criterion based on $p(\tilde{f}, \boldsymbol{X})$ (see [11]), and unfortunately, for the new selection criteria similar properties can be proved:

Corollary 3. *Assume that the inclusion functions of both the objective function and the constraint functions are isotone and have the zero convergence property. Consider the interval branch-and-bound algorithm that uses the feasibility test (and optionally other discarding tests) and that selects the next box to be subdivided by one of the following rules:*

Choose the box with the

- *maximal* $pup(f_k, \boldsymbol{Z})$ *value (Rule C5),*
- *maximal* $u(pup(f_k, \boldsymbol{Z}), -\underline{\boldsymbol{f}}(\boldsymbol{Z}))$ *value (utility function), or*
- *minimal* $pupb(f_k, \boldsymbol{Z})$ *value (Rule C7).*

Then both $\{\tilde{f}_k\}$ *and* $\{\underline{\boldsymbol{f}}(\boldsymbol{X}_k)\}$ *defined above may fail to converge to the global minimum value.*

Proof. As Theorem 1, Theorem 4 and Remark 4 show, algorithms using the investigated selection criteria may converge exclusively to one or more nonoptimal points (to an x' with $f(x') > f^*$). For such problems the sequence $\{\tilde{f}_k\}$ can converge to an $f(x')$ value or can remain a constant. The latter case may hold when an $\tilde{f}_k = f(x')$ is found in one of the first iteration steps, but later it cannot be updated due to the convergence of the method. If no cut-off test is applied then even $\tilde{f}_k < f(x')$ allows a convergence to x'. On the other hand, the convergent subsequences of $\{\underline{\boldsymbol{f}}(\boldsymbol{X}_k)\}$ converge exclusively to nonoptimal values due to the zero convergence property of $\boldsymbol{f}$. □

Nevertheless, there are several ways to obtain an updated estimate of the global minimum value, e.g. using random sampling of the search region, or running local search methods. A combination of the above techniques can give a good, converging estimate.

4.2. Convergence analysis of the new multisection rules

Results for adaptive multisection rules based on the index $p(\tilde{f}, \boldsymbol{X})$ were introduced in [4]. Remember, that $\tilde{f}$ means the updated best upper bound for the global minimum value while running the algorithm. In the following we assume that $\tilde{f}$ is the actual value of the sequence $\{\tilde{f}_k\}$ of (4) and we investigate the properties of the newly proposed index $pup(\tilde{f}, \boldsymbol{X})$.

Theorem 6. *Let us consider an interval B&B algorithm which selects the box with the smallest lower bound* $\underline{\boldsymbol{f}}(\boldsymbol{X})$ *as the next box to be subdivided. There exist optimization problems (1) for which the inclusion function* $\boldsymbol{f}(\boldsymbol{X})$ *of* $f(x)$ *is isotone and* α*-convergent and the inclusion functions* $\boldsymbol{g}_j(\boldsymbol{X})$ *of the constraint functions* $g_j(x)$ *are* α*-convergent, and the following statements are true:*

1. *An arbitrary large number* $N(>0)$ *of consecutive leading boxes selected by the algorithm have the properties that: neither of these processed boxes* $\boldsymbol{X}$ *contains a global minimizer point, and the related* $pup(\tilde{f},\boldsymbol{X})$ *values are larger than a preset* $P_2 < 1$.
2. *There exists a subsequence of the leading intervals converging to a global minimizer point, for which* $pup < P_1$ *(for a fixed* $0 < P_1 \le P_2$*).*

Proof. Consider an arbitrary optimization problem (1) that has two separate global minimizer points, x^* and x' $(f(x^*) = f^* = f(x'),\ x^* \neq x'$ and $x^*,\ x' \in \boldsymbol{X}_0,\ g_j(x^*) < 0,\ g_j(x') < 0,\ j = 1,\dots,r)$, and that has also a non-optimal feasible point $\hat{x} \in \boldsymbol{X}_0$ $(f(\hat{x}) > f(x^*),\ g_j(\hat{x}) < 0,\ j = 1,\dots,r)$. There exist obviously such problems.

Both statements of the theorem was proven in Theorem 1 in [4] for the unconstrained case and for $p(\tilde{f},\boldsymbol{X})$, by using a properly constructed interval inclusion function of the objective function. We extend this proof by defining suitable inclusion functions of the constraints.

1. Denote $\{\boldsymbol{X}_i\}_{i=0}^{N_0+N-1}$ the set of leading intervals generated by the algorithm for which $\hat{x} \in \boldsymbol{X}_i$ and after at most N_0 iterations $\hat{x}$ is separated from both global minimizers. (Notice that by the construction below this set will be well-determined.) First, let us use the usual natural interval extension providing the inclusion function $\boldsymbol{f}$ for each subbox of $\boldsymbol{X}_0$. This is an inclusion isotone and α-convergent $(\alpha = 1)$ inclusion function. Then for the elements of $\{\boldsymbol{X}_i\}_{i=0}^{N_0+N-1}$ redefine the inclusion function as

$$\boldsymbol{k}(\boldsymbol{Z}) = [\underline{\boldsymbol{f}}(\boldsymbol{Z}) - Dw(\boldsymbol{Z})^{\alpha}, \hat{f}],$$

where $\hat{f}$ is a value greater than or equal to $\overline{\boldsymbol{f}}(\boldsymbol{X}_0)$ and D is a constant large enough to satisfy the first statement of the theorem: the defined inclusion function will be α-convergent (with $\alpha = 1$) and inclusion isotone. Moreover, it guarantees that the first $N_0 + N$ leading intervals will be $\{\boldsymbol{X}_i\}_{i=0}^{N_0+N-1}$ and the $p(\tilde{f},\boldsymbol{Z})$ values on those intervals will be equal to 1 (notice that $\tilde{f} = \hat{f}$ holds in the first $N_0 + N$ iteration steps).

Finally, construct the inclusion functions of the constraints by natural interval extension. Due to the α-convergence of those inclusion functions and to the fact that $\hat{x}$ is strictly feasible, $pu(\boldsymbol{Z}) = pup(\tilde{f},\boldsymbol{Z}) = 1$ will hold for the elements of $\{\boldsymbol{X}_i\}_{i=0}^{N_0+N-1}$ if N_0 is large enough.

Consequently, the first $N_0 + N$ intervals selected by the algorithm will be $\{\boldsymbol{X}_i\}_{i=0}^{N_0+N-1}$, and the last N of them will not contain any global minimizers.

2. To prove the second statement of the theorem we extend the construction of the first part of the proof, namely, for intervals containing the global minimizer

point x' (but not containing $\hat{x}$) redefine the inclusion function of the objective function by

$$\boldsymbol{k}(\boldsymbol{Z}) = [\underline{f}(\boldsymbol{Z}) - Cw(\boldsymbol{Z})^{\alpha}, \overline{\boldsymbol{f}}(\boldsymbol{Z})],$$

where $f(\boldsymbol{Z})$ is the function range on $\boldsymbol{Z}$ (and then $\underline{f}(\boldsymbol{Z}) = f^*$), and the parameters C and α come from the α-convergence property of $\boldsymbol{f}$. Moreover, for intervals containing x^* (but containing neither $\hat{x}$ nor x') the inclusion of the objective function is given by

$$\boldsymbol{k}(\boldsymbol{Z}) = [\underline{f}(\boldsymbol{Z}) - Cw(\boldsymbol{Z})^{\alpha}, E],$$

with $E = \underline{\boldsymbol{k}}(\boldsymbol{Z}) + \delta(\tilde{f} - \underline{\boldsymbol{k}}(\boldsymbol{Z}))/P_1$ for a suitable constant $\delta > 1$. After a short computation one can easily see that the latter equality will imply $p(\tilde{f}, \boldsymbol{Z}) < P_1$ for such intervals. For this construction it can be shown that $\boldsymbol{k}$ is both inclusion isotone and α-convergent, and additionally, after a finite number of iteration steps (including those of the first $N_0 + N$ steps discussed above) the leading intervals will be exclusively the ones containing either x' or x^* due to the redefined lower bounds and the α-convergence of the inclusion.

To define suitable inclusion functions for the constraint functions, notice that arbitrary α-convergent inclusion functions (i.e. including the ones constructed by natural interval extension in the first part of our proof) are sufficient to prove statement 2: since x' and x^* are assumed to be strictly feasible, the α-convergence of the constraint functions implies $pu(\boldsymbol{Z}) = 1$ and then $pup(\tilde{f}, \boldsymbol{Z}) = p(\tilde{f}, \boldsymbol{Z})$ after a finite number of iteration steps for the leading intervals. Consequently, a subsequence of the leading intervals containing x^* can be chosen with $pup(\tilde{f}, \boldsymbol{Z}) = p(\tilde{f}, \boldsymbol{Z}) < P_1$. This proves 2. □

The conclusion of this theorem is that even with isotone and α-convergent inclusion function for the objective function and with α-convergent constraint inclusion functions, the generated sequence of the leading boxes can be distracted for an arbitrary long time (N iterations) from global minimizer points.

5. Computational studies

Most of the papers dealing with applications of interval analysis to global optimization consider unconstrained problems, and many papers dealing with constrained problems do not present computational experiments, except a few examples to show how the algorithm or a given step works. Some exceptions are [7,10,17,19,20,26,38]. In [28–30], Kearfott presented results for equality constrained problems, but his aim was to analyze different methods able to prove the existence of feasible points in a given box.

The numerical tests presented in this paper were carried out on a PC with an Intel Pentium IV 1400 MHz processor and with 1 Gbyte RAM running under the Linux operating system. The algorithms were based on the Toolbox for C-XSC [21,31] libraries, while the low-level interval-arithmetic routines were implemented by PROFIL/BIAS [32]. The Standard Time Unit (see e.g. [40]), the computation time needed to evaluate the Shekel-5 function 1000 times at the point (4,4,4,4) is 0.00076 seconds if we use a function with variables of type '`real`', and 0.00674 seconds if we use the corresponding natural interval extension (with variables of type '`interval`').

The feasibility test, the midpoint test and the monotonicity test were the only discarding tests used in the study. The working list was implemented by an AVL tree in which the elements were sorted according to the corresponding selection criterion.

5.1. Facility location problems

In this subsection we introduce an obnoxious facility location model presented in [17]. Its aim was to locate, within a geographical region, a facility considered undesirable by the inhabitants of the region, but which is not noxious for the health of the people, in the sense that its effects do not endanger peoples' lives, at least directly. The objective is to minimize the global repulsion of the inhabitants of the region to the location of the facility while taking environmental concerns into account which make some areas unsuitable for the location of the facility.

When the facility is located at the point x, the repulsion of the inhabitants at the city a_i is modeled by the function

$$\mathrm{rp}(a_i, x) = \frac{1}{1 + \exp(\alpha_i + \beta_i \cdot d_i(a_i, x))},$$

with $\beta_i > 0$, where $d_i(a_i, x)$ is a measure of the distance between a_i and x, and α_i and β_i are two parameters set for every existing city $a_i, i = 1, \ldots, m$. The lower the value of α_i, the higher the repulsion of the inhabitants to the location of the facility near their city or its outlying areas, and the higher the value of β_i, the faster is the change in the opinion from considering a distance non-acceptable to acceptable.

The objective of the model is to minimize the function

$$f(x) = \sum_{i=1}^{m} \omega_i \cdot \mathrm{rp}(a_i, x),$$

where ω_i is a positive weight proportional to the importance of the city a_i, usually related to its number of inhabitants.

Furthermore, around each city there is a forbidden area where the location is not allowed. This is to avoid the possibility of the facility being located in one of the cities or too close to them, since the minisum type objective function may cause such kind of result. Protected areas where the location is not allowed, and/or other areas where the location is not possible, are taken into account in the model by considering them as forbidden areas (sets of infeasible points). The forbidden areas can have any shape, the only requirement being to be able to specify the feasible set by a set of analytical constraints, $g_j(x) \leq 0,\ j = 1, \ldots, r$. Usually, the optimal location in these problems is on the border of the feasible set. That is why we have used this kind of test problems for the computational studies. For more details related with location aspects, the interested reader is referred to [16,33].

For our experiments we have generated a wide variety of problems according to the location model described above. The l_{2b}-norm, given by $l_{2b}(x) = \sqrt{b_1(x_1)^2 + b_2(x_2)^2}$, was used as distance function (see [18]); b_1 and b_2 are the parameters of the distance function considered in the study. The parameters of the model were drawn randomly from uniform distributions defined on the following intervals: $b_1, b_2 \in [1.0, 2.5]$ (the distance function was the same for every city), and similarly, $\alpha_i \in [-5, 1], \beta_i \in [1, 6]$ and $\omega_i \in [1, 10]$. The coordinates of the cities were drawn from a uniform distribution on the rectangle $([-1.0, 10.5], [-0.5, 10.0])$. Each city has a surrounding forbidden area whose size was proportional to its weight, specifically a circle with center a_i and radius of $\omega_i/20$. The protected areas were randomly chosen from 50 preset ones: 25 circles, 15 ellipses and 10 parabolas. The geographical regions were generated by adding to the initial box $\boldsymbol{X}_0$ some constraints from 50 preset ones: 25 linear constraints, 10 circles, 10 ellipses and 5 parabolas. The initial box $\boldsymbol{X}_0$ was always $([-2, 11], [-1, 10.5])$.

The number of cities considered was chosen as $n = 5, 10, 15, 20, 25, 45$ and the total number of constraints was $t = h + n$ (there are n forbidden circles around the cities) with $h = 5, 10, 20, 40, 60$, m of which were constraints of the geographical region and the rest, $h - m$, were protected areas (m was an integer randomly chosen between 0 and h). For each value of n and h, ten problems were generated by varying the values of the parameters, the coordinates of the cities and the constraints. In all, 300 problems were generated.

5.2. Standard global optimization problems

To extend the numerical investigations, we have generated constrained problems based on a wide standard test set of unconstrained problems used e.g. in [11, 15,35]. We have chosen 6 different types of hard problems from the above set, namely, the Shekel-10 (4-dimensional), the Hartman-6 (6 dim.), the Goldstein-Price (2-dim.), the Levy-3 (2-dim.), the Ratz-4 (2-dim.) and the EX2 (5-dim.) problems. Note, that the mentioned test set contains some other problems harder than the average, e.g. the Levy-8 to Levy-12, the Schwefel-2.7 and the Griewank-5 functions. For them one can see that $p(f^*, \boldsymbol{X}) = 0$ and then $pup(f^*, \boldsymbol{X}) = 0$ will hold for all boxes (this phenomenon was also discussed in [11]), thus, the interval selection rules C3 to C8 give the same result as C1, since in each step the basic decision criterion – which was C1 itself – is used. For this reason, the numerical investigations do not discuss the results on these functions.

For each problem selected we have produced 15 different constraint sets consisting of randomly generated linear and quadratic constraints. The number of constraints was $r = 15, 30$ and 45, and five different sets were generated for each r. The constraints were constructed in such a way that they have the global minimizer points of the test functions feasible (i.e. to keep the properties of the objective functions close to the minimizers even for the constrained cases).

5.3. Results for interval selection rules

Following [11], in our tests the approximation $f_k = f^*, k = 0, 1, \ldots$ was applied, i.e. the global optimum value was used as the f_k parameters. According to the theoretical results, this setting ensures convergence to a set of global optimizer points.

Since at the time of preparing the present manuscript no general methods producing estimates converging to f^* for rules C3 to C8 was available (cf. Corollary 3), in our numerical tests we needed the pre-given minimum value. However, this strong prerequisite is often fulfilled even for practical problems. For instance, in the case of solving the so-called circle packing problems [34] sometimes a very good approximation of the optimum value is at hand. Moreover, an initial function value is also known in advance when one wants to verify the validity of a given hypothetical optimum value, e.g. when the user wants to make sure that there exists no feasible solutions which attains, say 1% better objective function value than a known solution.

The termination criterion used by the algorithms was $w(\boldsymbol{f}(\boldsymbol{X})) < \epsilon$ for a candidate box $\boldsymbol{X}$. The ϵ parameter was set to 10^{-4} for the whole numerical study. This stopping criterion is a consequence of the assumption $f_k = f^*$: it is easy to see that if our objective is the exploration of the whole search space and if f^* is used in the cut-off test from the beginning, then the generated B&B tree will be independent from the interval selection rule. However, the order of processing the particular subboxes will still depend on the selection rule, and therefore, it is apparent to test the algorithm variants e.g. with a stopping criterion above. On the other hand, this criterion still allows us to provide guaranteed enclosures both for the global minimum and for all the minimizers (by taking the boxes remained in $\mathcal{L}_\mathcal{W}$ also into account).

In case of the C4, C6 and C8 hybrid rules, the algorithm parameter N_m was set to 100, thus, at the first phase at each iteration level the algorithm was dealing only with the 100 most promising boxes. The rest of the boxes were processed using the traditional C1 selection criterion at the end of the algorithm.

In our first test concerning the subinterval selection rules every problem was solved with the rules C1-C8 introduced in Section 2 and Section 3. We refer the algorithms using one of the selection rules by simply using the notation of the selection rule. Keep in mind, that each selection rule can be combined with different static subdivision rules (and this is also necessary in order to separate the effect of the subinterval selection from that of the multisection). We have tested the static bisection/multisection rules defined at the end of Section 3. For the whole set of problems (including the facility location and standard problems) the static tetrasection proved to be the best to combine with all of the interval selection rules. Thus, we demonstrate the numerical results achieved by the static tetrasection.

The computational results obtained solving the 300 facility location problems are summarized in Table 1. The total values of CPU time (CPUt, in seconds), sum of the maximum number of boxes stored in the working list(s) (ML), the number of objective function evaluations (NFE) and the number of objective

Rule	CPUt	rel.	ML	rel.	NFE	rel.	NGE	rel.
C1	289.15		5,962		56,639		21,743	
C2	322.57	112%	5,057	85%	64,488	114%	24,947	115%
C3 (pf)	75.27	26%	2,794	47%	17,412	31%	3,780	17%
C4 (h-pf)	295.38	102%	5,430	91%	60,311	106%	22,572	104%
C5 (pup)	110.31	38%	3,496	59%	22,956	41%	6,383	29%
C6 (h-pup)	297.28	103%	5,455	91%	60,651	107%	22,739	105%
C7 (pupb)	105.14	36%	3,969	67%	21,480	38%	5,381	25%
C8 (h-pupb)	300.05	104%	5,478	92%	60,664	107%	22,736	105%

Table 1. Result of solving 300 facility location problems. The table includes total running time in seconds (CPUt), sum of maximal list lengths (ML) and total number of objective function (NFE) and gradient (NGE) evaluations for 8 algorithm versions. Each column contains also relative values compared to rule C1.

gradient evaluations (NGE) are given. For each quantity we also give its relative value compared to that of the traditional Moore-Skelboe algorithm denoted by C1. Summarizing the results of Table 1 regarding the running time, the C3, C5 and C7 non-hybrid methods proved to work far quicker than C1. The best method is based on the $p(f_k, \boldsymbol{Z})$ index: this algorithm required 26% of the CPU time used by C1, while C7 (based on $pupb(f_k, \boldsymbol{Z})$) and C5 (based on $pup(f_k, \boldsymbol{Z})$) required a little more, 36% and 38%, respectively. On the other hand, the differences between the algorithms with the hybrid rules C4, C6, C8 and the variant C1 were a few percentages.

Investigating the storage complexity of the algorithms, we found that every rule causes smaller memory requirements than the classical Moore-Skelboe method. Rule C3 is the best choice with 47%, followed by the other non-hybrid rules, C5 (59%) and C7 (67%). The hybrid rules required 8-9% less memory than C1.

The relative values of objective function and gradient evaluations show almost the same pattern as that of the CPU times. Consequently, reduced running time can be reached by reducing the function evaluations, and moreover, the list-handling operations have no significant effect to the running time (since the memory requirements needed to solve these type of problems are small). Again, the algorithm with rule C3 computed the function and gradient values the smallest number of times (31% for NFE and 17% for NGE), rules C5 and C7 were also working very efficiently, while the C4, C6 and C8 rules resulted in a little more evaluations than the basic rule, 6-7% more than C1 for NFE and 4-5% more than C1 for NGE.

The hardness of the discussed set of facility location problems could be expressed by the NFE and NGE values: the 300 problems were solved in the basic cases by 56,639 function evaluations (in average 189 evaluations) and 21,743 gradient evaluations (in average 72 evaluations). As we will see, the constrained versions of the standard optimization problems proved to be harder than facility problems (and were sometimes very hard to solve).

Consider now the results achieved by solving 15 Shekel-10 problems with the generated constraints (Table 2). Investigating the running time we can state that all the heuristic algorithm methods proved to work quicker than the C1

Rule	CPUt	rel.	ML	rel.	NFE	rel.	NGE	rel.
C1	3.20		31		2,618		1,534	
C2	3.20	100%	31	100%	2,618	100%	1,534	100%
C3 (pf)	3.07	96%	33	106%	2,523	96%	1,529	100%
C4 (h-pf)	3.11	97%	31	100%	2,618	100%	1,534	100%
C5 (pup)	3.06	96%	33	106%	2,523	96%	1,529	100%
C6 (h-pup)	3.11	97%	31	100%	2,618	100%	1,534	100%
C7 (pupb)	3.13	98%	31	100%	2,618	100%	1,534	100%
C8 (h-pupb)	3.12	98%	31	100%	2,618	100%	1,534	100%

Table 2. Result of solving 15 constrained Shekel-10 problems.

Rule	CPUt	rel.	ML	rel.	NFE	rel.	NGE	rel.
C1	35,022		2,559,703		25,293,057		3,761,838	
C2	31,683	90%	1,436,597	56%	26,102,840	103%	3,911,177	104%
C3 (pf)	2,866	8%	712,566	28%	2,138,638	8%	272,999	7%
C4 (h-pf)	195	1%	31,402	1%	149,647	1%	32,913	1%
C5 (pup)	3,280	9%	225,776	9%	2,515,313	10%	368,425	10%
C6 (h-pup)	192	1%	22,453	1%	144,329	1%	39,133	1%
C7 (pupb)	32,986	94%	6,636	0%	26,102,840	103%	3,911,177	104%
C8 (h-pupb)	46,729	133%	2,558,641	100%	25,293,094	100%	3,761,838	100%

Table 3. Result of solving 15 constrained Hartman-6 problems.

variant, although, the difference between the results is only some percentages. Rule C3 and the newly proposed rule C5 were the quickest variants requiring 4% less CPU time than C1. The algorithms with the latter two rules have also the smallest NFE values (again, 4% less than C1). The sums of the maximal lengths of the working lists show very low values for all interval selection criteria. All the problems were solved using 2 or 3 list elements, which means that already the basic method of our algorithm variants is sophisticated enough to avoid unnecessary box-splittings for the Shekel-10 problems.

Table 3 shows the test results of our algorithms executed on the constrained Hartman-6 problem. As we found, the additional constraints made this problem very hard. Rules C4 and C6 proved to give the most efficient methods, requiring two orders of magnitude less running time, memory requirement, function and gradient evaluations compared to C1. For the algorithms using C4 to C6, the $pup(f_k, \boldsymbol{Z})$-based C6 method has 2% and 4% improvement on CPUt and NFE, respectively, but it resulted 16% more gradient evaluations than C4. This fact, i.e. the increased number of NGE value for methods using constraints information can be observed also for other test problems. The reason for it is that the gradient is evaluated only for feasible boxes when the monotonicity test is applied. If two boxes have closely the same $p(f_k, \boldsymbol{Z})$ value and the first one is feasible but the other is undetermined, then the rules C5 to C8 may better select the feasible box for subdivision than the heuristics C3 and C4.

Returning back to the Hartman-6 problems, it is worth to mention that C6 used 28% less memory than C4, which is very promising for the new rules. Investigating the other rules we can state that C3 and C5 were also much more efficient than C1: they both produced 90-93% less CPU time, NFE and NGE

Rule	CPUt	rel.	ML	rel.	NFE	rel.	NGE	rel.
C1	79.51		24,188		423,612		323,000	
C2	75.91	95%	30,979	128%	423,717	100%	323,105	100%
C3 (pf)	1.37	2%	2,519	10%	7,833	2%	4,055	1%
C4 (h-pf)	18.31	23%	14,035	58%	100,896	24%	67,079	21%
C5 (pup)	1.13	1%	1,976	8%	6,415	2%	3,423	1%
C6 (h-pup)	18.14	23%	13,442	56%	99,698	24%	67,927	21%
C7 (pupb)	76.73	97%	18,984	78%	414,548	98%	319,840	99%
C8 (h-pupb)	78.03	98%	23,145	96%	423,612	100%	323,000	100%

Table 4. Result of solving 15 constrained Goldstein-Price problems.

values (showing a small advantage of C3 against C5). The ML quantities show bigger difference: 9% against 28%, which means 68% of relative improvement of C5 compared to C3. Another interesting issue is the memory requirement of the C7 method: although it produced only 6% less CPU time than C1, it worked with far the smallest lists among all the selection rules. Thus, in cases when the storage is limited and the problem looks to be hard to solve, the new C7 method could also be suggested.

When analyzing the numerical results achieved for the Goldstein-Price problems (Table 4), first of all we must emphasize the efficiency of the C3 and C5 rules compared to the basic C1 rule: 1-2% of the running time, objective function and gradient evaluations, and 8-10% for storage complexity. Moreover, our newly proposed C5 rule (based on $pup(f_k, \boldsymbol{Z})$) has improved the efficiency of rule C3 even in this case: algorithm C5 required 18% less CPU time, 22% less memory for the working list, and evaluated the objective and the gradient 18% and 16% less times than C3, respectively. Rules C4 and C6 behave similarly compared to the Moore-Skelboe rule: they need 23% for the CPU time and 56-58% for the memory requirement. The other variants, namely C2, C7 and C8 produced up to 5% of improvements in the CPUt, NFE and NGE quantities. However, it is worth to mention that the method based on C7 required again far less memory than the other $pupb(f_k, \boldsymbol{Z})$-based method (C8): only 78% instead of 96% of that of C1.

Table 5 contains test results produced on the Levy-3 problem variants. In terms of the processor time needed, the number of function evaluations and the number of gradient evaluations the algorithm variants resulted in similar, low quantities. Both C3 and C5 used 45% of the running time of C1 and achieved 46% and 47-49% in the NFE and NGE values, respectively, while the differences between the other methods and C1 were negligible. Nevertheless, the ML values show similar results as those observed for the Hartman-6 problem. Namely, the method with the new C7 rule used again the smallest amount of memory (40%) and also a newly proposed variant, C5 was the second best (57%). Note that the memory requirement needed to solve Levy-3 instances was much smaller than for the Hartman-6 problems.

One can observe several similarities between the results of the Ratz-4 function (shown in Table 6) and the Levy-3 function. Again, two non-hybrid rules, C3 and C5 were the most efficient (15-16% in CPUt, 16% in NFE and 12-16% in NGE).

Rule	CPUt	rel.	ML	rel.	NFE	rel.	NGE	rel.
C1	2.11		276		5,006		2,964	
C2	2.08	99%	277	100%	5,003	100%	2,961	100%
C3 (pf)	0.96	45%	232	84%	2,315	46%	1,380	47%
C4 (h-pf)	2.10	100%	300	109%	5,006	100%	2,964	100%
C5 (pup)	0.96	45%	156	57%	2,303	46%	1,467	49%
C6 (h-pup)	2.10	100%	297	108%	5,006	100%	2,964	100%
C7 (pupb)	2.12	100%	110	40%	5,004	100%	2,962	100%
C8 (h-pupb)	2.12	100%	268	97%	5,004	100%	2,962	100%

Table 5. Result of solving 15 constrained Levy-3 problems.

Rule	CPUt	rel.	ML	rel.	NFE	rel.	NGE	rel.
C1	1.41		790		12,036		6,998	
C2	1.48	105%	618	78%	12,672	105%	7,163	102%
C3 (pf)	0.23	16%	389	49%	1,866	16%	853	12%
C4 (h-pf)	1.50	106%	718	91%	12,582	105%	7,073	101%
C5 (pup)	0.21	15%	366	46%	1,982	16%	1,097	16%
C6 (h-pup)	1.53	109%	678	86%	12,582	105%	7,073	101%
C7 (pupb)	1.51	107%	206	26%	12,920	107%	7,367	105%
C8 (h-pupb)	1.56	111%	554	70%	12,920	107%	7,367	105%

Table 6. Result of solving 15 constrained Ratz-4 problems.

The other rules were slower by 5% to 11% than C1, and also evaluated the objective function 5-7% more times. In term of the storage complexity, again C7 was the best choice followed by C5 (26% and 46%), and additionally, the algorithm using the C1 rule was improved by all the other variants.

As a brief conclusion of our investigations on interval selection rules we can state that the new type of methods (C5 to C8) may improve the efficiency of the $p(f_k, \boldsymbol{Z})$-type rules by using constraint information. This improvement can be achieved even in the cases when the problem is hard to solve and a rule using $p(f_k, \boldsymbol{Z})$ alone is also very efficient, e.g. on problems Goldstein-Price and Hartman-6. Figure 1 shows a positive example: the splittings made by the algorithm variant C4 (left) and C6 (right) are represented for one of the constrained Goldstein-Price problems in an area close to the global minimizer located at $(0, -1)$. The relevant constraints are also included in the figures. As the left figure shows, the application of the $p(f_k, \boldsymbol{Z})$ values (not taking the constraints into account) resulted in a lot of unnecessary splittings on the border of the feasible set. These splittings are avoided when using rule C6, since the pu values force the algorithm to split undetermined boxes on the border less probably, instead, split better feasible boxes closer to the global minimizer.

The effect of the additional constraint information can be observed in both the reduced CPU time together with the smaller number of function evaluations, and mainly in the memory usage: for the problems of Hartman-6, Goldstein-Price, Levy-3 and Ratz-4 (i.e. again for the harder ones) the methods using the $pup(f_k, \boldsymbol{Z})$ or $pupb(f_k, \boldsymbol{Z})$ variants were better than the corresponding $p(f_k, \boldsymbol{Z})$-based rules, and in three cases Rule C7 uses the smallest amount of memory. Nevertheless, it is worth to mention that although we have chosen different

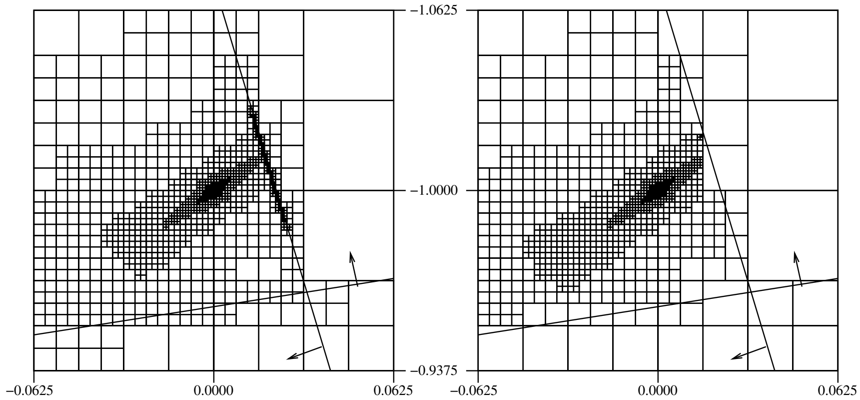

Fig. 1. Splittings on a constrained Goldstein-Price problem close to the global minimizer point by the algorithm using C4 (left) and using C6 (right).

types of problems, the test set is limited and even on this set the efficiency of the rules shows a big variety. For instance, in many cases the non-hybrid variants C3 and C5 (in one case also C7) are significantly better than hybrid rules C4, and C6 (for facility location, Goldstein-Price, Levy-3 and Ratz-4), but on the hardest Hartman-6 problem the hybrid-type rules were more efficient. From the numerical experiences of [6] we know that for unconstrained problems the efficiency of a $p(f_k, \boldsymbol{Z})$ index based hybrid method is very sensitive to the N_m value. Carrying out a study on this effect for constrained problems and for the *pup* and *pupb* indices could be a possible future work.

The most important results were obtained for the very hard to solve EX2 test problem [15]. This time no algorithm variant was able to solve all the 15 constrained problems, i.e. for all the studied methods there were problems that could not be solved having the usual stopping condition. Obviously, with larger ϵ values more problem instances could be solved. The C1 and C8 rules allowed to solve only 2 problem instances, while the new rules of C5 and C7 resulted in 10 and 9 solved cases, respectively (for rules C2, C3, C4 and C6 the related numbers were 3, 6, 4, and 4, respectively). Although for this test problem the earlier comparison tables with the efficiency indicators does not make sense (since the solved problem instances were usually different), for the user it is more important to know which algorithm version could solve more problem instances.

This result has the message that for difficult to solve problems, the new constrained optimization techniques have a clear advantage: the C5 and the C7 interval selection rules made the algorithm capable to solve the by far the most test problem instances. Since the reliable solution of global optimization problems is regarded as very difficult even for the low dimensional cases, the new methods can provide effective tools for such problems.

Rule	CPUt	rel.	ML	rel.	NFE	rel.	NGE	rel.
S1_/2	394.65		6,643		77,981		21,854	
S2_/4	289.15	73%	5,962	90%	56,639	73%	21,743	99%
S3_/9	321.08	81%	5,445	82%	62,748	80%	32,738	150%
S4_pf_/2,4,9	277.25	70%	5,728	86%	54,606	70%	25,385	116%
S5_pup_/2,4,9	291.79	74%	5,773	87%	55,892	72%	24,738	113%

Table 7. Result of solving 300 facility location problems. Each column contains also relative values compared to rule S1_/2.

5.4. Results for adaptive multisection rules

Several static and adaptive multisection methods [4,9,35] were also tested on the above described large test set of facility location problems. Each problem was solved five times, using the following subdivision rules:

S1_/2: classical static bisection by the widest interval.
S2_/4: static tetrasection (bisection in the two widest directions).
S3_/9: static multisection into 9 subboxes (trisection in the two widest directions).
S4_pf_/2,4,9: adaptive multisection based on $p(\tilde{f}, \boldsymbol{X})$ using the bisection, the tetrasection and the multisection into 9 subboxes as possible subdivision types.
S5_pup_/2,4,9: adaptive multisection based on $pup(\tilde{f}, \boldsymbol{X})$ as in S4.

Since the classic Moore-Skelboe interval selection rule was applied for each method, the static rules were exactly the same algorithms as presented in the previous subsection under notations C1_/2, C1_/4, C1_/9, respectively.

Earlier we pointed out that the key property which determines the running time is the number of function evaluations. In order to find good $P1$ and $P2$ threshold values for the adaptive rules, a sample set containing 30 out of the 300 facility location problems was generated (by choosing one problem randomly for each value of n and h, see subsection 5.1). We considered the total number of function evaluations on that sample set as the objective function to be minimized (for the variables $P1$ and $P2$). Following [4], the stochastic global optimization algorithm called SASS was used to solve this optimization problem. We have obtained $P1 = 0.0,\ P2 = 0.278475$ for the adaptive multisection based on $p(\tilde{f}, \boldsymbol{X})$, and $P1 = 0.0,\ P2 = 0.199556$ for the rule based on the $pup(\tilde{f}, \boldsymbol{X})$ index.

Tables 7-9 shows the results obtained. The tables are organized in the same way as for the interval selection rules. From Table 7 we can conclude that the original adaptive rule S4 can be preferred, but it required only a slightly less CPU time than S5 (70% against 74%). S4 was also better in the terms of storage complexity (86% against 87%), and the number of function evaluations (70% against 72%). Notice that taking the constraints into account in the heuristic index results in a better choice for gradient evaluations: the NGE value was 113% for rule S5 and 116% for rule S4 compared to the classical S1 rule, respectively.

Rule	IT	rel.	NBG	rel.
S1_/2	31,968		64,236	
S2_/4	14,489	45 %	57,390	89 %
S3_/9	8,499	27 %	75,267	117 %
S4_pf_/2,4,9	10,741	34 %	62,833	98 %
S5_pup_/2,4,9	11,591	36 %	59,010	92 %

Table 8. Result of solving 300 facility location problems. The table includes sum of the necessary iterations (IT) and the sum of the number of generated boxes (NBG).

Rule	FT	rel.	MPT	rel.	MT	rel.	Sum	rel.
S1_/2	7,903	30%	10,918	41%	7,773	29%	26,594	100%
S2_/4	11,608	31%	16,684	45%	9,151	24%	37,443	100%
S3_/9	18,989	31%	29,973	49%	12,050	20%	61,012	100%
S4_pf_/2,4,9	15,768	34%	18,294	39%	12,577	27%	46,639	100%
S5_pup_/2,4,9	11,680	28%	19,211	46%	11,085	26%	41,976	100%

Table 9. Result of solving 300 facility location problems. The table includes the total number of boxes eliminated by the feasibility test (FT), by the midpoint and cut-off test (MPT) and by the monotonicity test (MT), respectively. The relative figures are now calculated on the basis of the row sums.

Table 8 contains additional efficiency indicator information. According to these less important factors, the multisection algorithms S3 to S5 need less iterations than S1 and S2. Moreover, as compared to the ML values one can conclude that a smaller iteration number resulted smaller working lists for each variant. The number of generated boxes shows that S5_pup_/2,4,9 generated less boxes than both the basic, bisection algorithm variant and the S4 adaptive multisection rule, but the S2_/4 rule was even better from this point of view.

Table 9 indicates how many subboxes were deleted by the applied accelerating devices. According to these figures, for each rule the cut-off test is the most effective, followed by the feasibility test and the monotonicity test. However, we can conclude that even the most expensive monotonicity test (applied only after the other discarding tests if needed) played an important role in our algorithms.

6. Conclusions and future research

According to the theoretical results the new interval selection rules enable the branch-and-bound interval optimization algorithm to converge to a set of global optimizer points assumed we have a proper sequence of $\{f_k\}$ parameter values. The convergence properties obtained were very similar to those proven for the unconstrained case, and they give a firm basis for computational implementation.

Through an extensive computational study, it has been shown that the constrained version interval selection rules and to a less extent also the new adaptive multisection rules have several advantageous features that can contribute to the efficiency of the interval optimization techniques. Although these conclusions were in part obtained for a particular obnoxious facility location model and for standard global optimization test problems, they may be valid for other constrained problems for which the optimal solution is close to the border of the feasible set, since for such problems many undetermined boxes may have to be checked.

A direction for future research is to investigate whether other sequences for the multisection type, different from the sequences $(2, 4, 9)$ used in this paper (strategies S4 and S5, respectively), may produce better results (e.g. the sequences $(2, 9, 16)$ or $(2, 9, 25)$ could be good options). The study of multisection using three thresholds, thus following sequences with four components deserve attention as well as multisection performing a different number of cuts depending on the coordinate direction chosen (in particular, multisection which choose the number of cuts to be performed orthogonal to direction i depending on the value of the rule $D(i)$ used for the selection of subdivision directions, as for instance Rules A, B and C in [14,15,41,42]). Another area for future research is to find additional theoretical convergence results for the new adaptive multisection rule, and to find a way to produce the $\{f_k\}$ sequence converging to f^* — without a priori knowledge.

Acknowledgements

This work was supported by the Grants FKFP 0449/99, OTKA T 32118 and T 34350, by the Ministry of Education of Spain (CICYT TIC2002-00228), by the Ministry of Science and Technology of Spain (under the research project BEC2002-01026, in part financed by the European Regional Development Fund), and by the Hungarian-Spanish Bilateral Collaboration Project SP-25/01.

References

1. S. Berner (1996), New results on verified global optimization, *Computing* 57, 323–343.
2. L.G. Casado (1999), Optimización global basada en aritmética de intervalos y ramificación y acotación: paralelización, Ph.D. Thesis, University of Málaga, Spain (in Spanish).
3. L.G. Casado and I. García (1998), New load balancing criterion for a parallel interval global optimization algorithms, Proceedings of the 16th IASTED International Conference, 321–323, Garmisch-Partenkirchen, Germany.
4. L.G. Casado, I. García, and T. Csendes (2000), A new multisection technique in interval methods for global optimization, *Computing* 65, 263–269.
5. L.G. Casado, I. García, and T. Csendes (2001), A heuristic rejection criterion in interval global optimization algorithms, *BIT* 41, 683–692.
6. L.G. Casado, J.A. Martínez, and I. García (2001), Experimenting with a new selection criterion in a fast interval optimization algorithm, *Journal of Global Optimization* 19, 247–264.
7. A.E. Csallner (1993), Global optimization in separation network synthesis, *Hungarian Journal of Industrial Chemistry* 21, 303–308.
8. A.E. Csallner and T. Csendes (1996), On the convergence speed of interval methods for global optimization, *Computers, Mathematics and Applications* 31, 173–178.
9. A.E. Csallner, T. Csendes, and M.Cs. Markót (2000), Multisection in interval branch–and–bound methods for global optimization I. Theoretical results, *Journal of Global Optimization* 16, 371–392.
10. T. Csendes (1998), Optimization methods for process network synthesis — a case study. In Christer Carlsson and Inger Eriksson (eds.): Global and multiple criteria optimization and information systems quality, Abo Academy, Turku, 113–132.

11. T. Csendes (2001), New subinterval selection criteria for interval global optimization, *Journal of Global Optimization* 19, 307–327.
12. T. Csendes, R. Klatte, and D. Ratz (2000), A posteriori direction selection rules for interval optimization methods, *Central European Journal of Operations Research* 8, 225–236.
13. T. Csendes T. and J. Pintér (1993), The impact of accelerating tools on the interval subdivision algorithm for global optimization, *European Journal of Operational Research* 65, 314–320.
14. T. Csendes and D. Ratz (1996), A review of subdivision direction selection in interval methods for global optimization, *ZAMM* 76, 319–322.
15. T. Csendes and D. Ratz (1997), Subdivision direction selection in interval methods for global optimization, *SIAM Journal on Numerical Analysis* 34, 922–938.
16. Z. Drezner (1995), Facility location: a survey of applications and methods, Springer-Verlag, Berlin.
17. J. Fernández, P. Fernández, and B. Pelegrín (2000), A continuous location model for siting a non-noxious undesirable facility within a geographical region, *European Journal of Operational Research* 121, 259–274.
18. J. Fernández, P. Fernández, and B. Pelegrín (2002), Estimating actual distances by norm functions: a comparison between the $l_{k,p,\theta}$-norm and the $l_{b_1,b_2,\theta}$-norm and a study about the selection of the data set, *Computers and Operations Research* 29, 609–623.
19. J. Fernández and B. Pelegrín (2001), Using interval analysis for solving planar single-facility location problems: new discarding tests, *Journal of Global Optimization* 19, 61–81.
20. A. Goos and D. Ratz (1997), Praktische Realisierung und Test eines Verifikationsverfahrens zur Lösung globaler Optimierungsprobleme mit Ungleichungsnebenbedingungen, Forschungsschwerpunkt Computerarithmetik, Intervallrechnung und Numerische Algorithmen mit Ergebnisverifikation, technical report (available at `http://www.uni-karlsruhe.de/~iam/html/reports.html`).
21. R. Hammer, M. Hocks, U. Kulisch, and D. Ratz (1995), C++ Toolbox for verified computing, Springer-Verlag, Berlin.
22. E. Hansen (1980), Global optimization using interval analysis – the multidimensional case, *Numerische Mathematik* 34, 247–270.
23. E. Hansen (1992), Global optimization using interval analysis, Marcel Dekker, New York.
24. R. Horst and P.M. Pardalos (eds.) (1995), Handbook of global optimization, Kluwer, Dordrecht.
25. R. Horst and H. Tuy (1996), Global optimization. Deterministic approaches, Springer-Verlag, Berlin.
26. J.Z. Hua, J.F. Brennecke, and M.A. Stadtherr (1998), Enhanced interval analysis for phase stability: cubic equation of state models, *Industrial Engineering Chemical Research* 37, 1519–1527.
27. R.B. Kearfott (1996), Rigorous global search: continuous problems, Kluwer, Dordrecht.
28. R.B. Kearfott (1996), Test results for an interval branch and bound algorithm for equality-constrained optimization, in: State of the art in global optimization, Kluwer, Dordrecht, 181–199.
29. R.B. Kearfott (1996), On verifying feasibility in equality constrained optimization problems, technical report (available at `http://interval.louisiana.edu/preprints.html`).
30. R.B. Kearfott (1998), On proving existence of feasible points in equality constrained optimization problems, *Mathematical Programming* 83, 89–100.
31. R. Klatte, U. Kulisch, A. Wiethoff, C. Lawo and M. Rauch (1993), C-XSC – A C++ class library for extended scientific computing, Springer-Verlag, Heidelberg.
32. O. Knüppel, (1994), PROFIL/BIAS — a fast interval library, *Computing* 53, 277–287.
33. R.F. Love, J.G. Morris, and G.O. Wesolowsky (1988), Facilities location: models and methods, North-Holland, New York.
34. M.Cs. Markót (2000), An interval method to validate optimal solutions of the "packing circles in a unit square" problems, *Central European Journal of Operations Research* 8, 63–78.
35. M.Cs. Markót, T. Csendes, and A.E. Csallner (1999), Multisection in interval branch-and-bound methods for global optimization. II. Numerical tests, *Journal of Global Optimization* 16, 219–228.
36. H. Ratschek and J. Rokne (1984), Computer methods for the range of functions, Ellis Horwood, Chichester.
37. H. Ratschek and J. Rokne (1988), New computer methods for global optimization, Ellis Horwood, Chichester.

38. H. Ratschek and J. Rokne (1993), Experiments using interval analysis for solving a circuit design problem, *Journal of Global Optimization* 3, 501–518.
39. H. Ratschek and R.L. Voller (1991), What can interval analysis do for global optimization?, *Journal of Global Optimization* 1, 111–130.
40. D. Ratz (1994), Box-splitting strategies for the interval Gauss-Seidel step in a global optimization method, *Computing* 53, 337–353.
41. D. Ratz (1996), On branching rules in second-order branch-and-bound methods for global optimization, in: G. Alefeld, A. Frommer, and B. Lang (eds.), Scientific computing and validated numerics 221–227, Akademie-Verlag, Berlin.
42. D. Ratz and T. Csendes (1995), On the selection of subdivision directions in interval branch-and-bound methods for global optimization, *Journal of Global Optimization* 7, 183–207.
43. R. Vaidyanathan and M. El-Halwagi (1994), Global optimization of nonconvex nonlinear programs via interval analysis, *Computers & Chemical Engineering* 18, 889–897.