Journal of Global Optimization **7:** 297–331, 1995.

A Branch and Bound Algorithm for Bound Constrained Optimization Problems without Derivatives

CHRISTIAN JANSSON and OLAF KNÜPPEL
Technische Informatik III, TU Hamburg-Harburg, Eissendorferstrasse 38, 21071 Hamburg-Hamburg, Germany

(Received: 12 May 1993; accepted: 31 March 1995)

Abstract. In this paper, we give a new branch and bound algorithm for the global optimization problem with bound constraints. The algorithm is based on the use of inclusion functions. The bounds calculated for the global minimum value are proved to be correct, all rounding errors are rigorously estimated. Our scheme attempts to exclude most "uninteresting" parts of the search domain and concentrates on its "promising" subsets. This is done as fast as possible (by involving local descent methods), and uses little information as possible (no derivatives are required). Numerical results for many well-known problems as well as some comparisons with other methods are given.

Mathematics Subject Classifications (1991) 49D37, 65G10.

Key words: Global optimization, interval arithmetic.

1. Introduction

Global optimization is calculating the global optimum of an objective function over a set of feasible points. In this paper we consider the following global optimization problem

$$\text{Min}\{ f(x) \mid x \in X \}, \quad X := \{ x \in \mathbb{R}^n \mid \underline{X} \leq x \leq \overline{X} \}, \tag{1}$$

where $f : X \to \mathbb{R}$, and the *set of feasible points* X is a *box* or *interval vector* with $\underline{X} \leq \overline{X}$, $\leq$ is to be understood componentwise. The *global minimum* (if it exists) is denoted by $f^* := \text{Min}\{ f(x) \mid x \in X \}$, and the set of *global minimum points* is denoted by

$$X^* := \{ x^* \in X \mid f(x^*) = f^* \}. \tag{2}$$

During the last two decades several methods have been developed for solving global optimization problems. Branch and bound schemes have been recognized as deterministic methods that calculate bounds for f^* and/or X^*. Their main components are (i) techniques for partitioning the set of feasible points X into subregions Y, (ii) the calculation of bounds for the range of f on those subregions, and (iii) techniques to discard some of the subregions.

One of the most important aspects of those methods is the calculation of bounds for the range of functions. To our knowledge, Moore [21] was the first to discover that interval arithmetic allows computing rigorous bounds for the range of a function over a box X, where the function is given by an arithmetical expression. Based on this results a branch and bound strategy with some of Moore's principles was given by Skelboe [34] and improved by Moore [22]. Important modifications of Moore's method for solving the global optimization problem are due to Hansen [8], [9], [10]. Especially, he gives substantial attention to problems where additionally rigorous bounds for the range of the first and second derivative are available, and proposed improved versions using interval Newton methods, monotonicity tests, and nonconvexity tests. A detailed convergence analysis of those methods was first given by Ratschek [28]. A special method for solving Minimax Problems by using interval arithmetic can be found in [33]. Two excellent treatments of how to apply interval methods to nonlinear systems and of global optimization problems are given by Ratschek & Rokne [30] and Hansen [10]. These books contain many references which are related to interval arithmetic, nonlinear systems, and global optimization. For other branch and bound methods, not using interval arithmetic, the reader is referred to [11], [27], [35].

The components (i) and (iii) of a branch and bound method have important influence on the efficiency and the storage requirements. A common technique for discarding subregions is the following: a subregion Y contains no global minimum point and can be discarded, if a point $x \in X$ is known such that the calculated lower bound of f on a subregion Y of X is greater than $f(x)$. Hence, for discarding subregions and accelerating branch and bound schemes it would be best if f^* would be known at the very beginning.

In contrast to other branch and bound techniques described in the literature mentioned above, the goal of our paper is to consider a branch and bound algorithm which incorporates local optimization algorithms for computing approximations of f^* and X^* at the very beginning. There are two important difficulties by using local optimization algorithms for global optimization problems. First, local optimization algorithms rely heavily on the starting point, and the region of attraction for a global minimum point or a stationary point may be very small. Secondly, it is difficult when to call a local optimization algorithm in a global optimization method. Ideally, the local optimization algorithm should **only** be called when an approximation of a global minimum point will actually be computed.

In our method we use *inclusion functions* which give rigorous bounds for the range of values of functions. One way to obtain inclusion functions is to use the tools of interval arithmetic, where rounding errors can rigorously be estimated. For example, functions which are given by arithmetic expressions, and which additionally may involve standard functions, an inclusion function easily can be obtained in the following way: The variables, real operations, and standard functions are replaced by interval variables, real interval operations, and interval standard functions, respectively. In the last two decades, methods for calculating inclusion func-

tions are given for many problems, like linear and nonlinear equations, eigenvalue problems, differential equations, integral equations, etc. If the objective function is implicitely defined by such a problem, we may calculate with those methods a corresponding inclusion function. On the other hand, our method cannot be applied to problems where bounds for the range of the objective function are not available; for example, functions which are computed by complex routines that only calculate function evaluations at trial points.

We assume that the reader is familiar with the elementary concepts of interval arithmetic. These concepts and their applications are described in the monographs Alefeld & Herzberger [1], Kulisch & Miranker [19], Moore[23], Neumaier [26], and Ratschek & Rokne [29]. These books contain many examples.

In our scheme, we use inclusion functions for calculating bounds for the range on subregions, and also for incorporating local optimization algorithms. In many experiments we observed that usually inclusion functions overestimate the true range of the objective function, but they have the following property: if lower bounds of f on two subboxes with equal diameter are calculated, then in many cases the subbox with the smaller lower bound contains smaller function values. Our branch and bound scheme is motivated by this observation. This scheme improves starting points for the local optimization algorithm and attempts to avoid the difficulties mentioned above.

Our method calculates approximations and guaranteed bounds of the global minimum value and the global minimum points. The bounds calculated for the global minimum points are rough compared to the bounds for the global minimum value. Neither derivatives of the objective function nor derivatives of the corresponding inclusion function are required.

The paper is organized as follows. Section 2 describes our branch and bound method for solving the bound constrained optimization problem. In Section 3 a detailed convergence analysis of the method is given. Section 4 contains numerical results of 22 test problems, and in Section 5 some conclusions are given. We mention that our method is faster than many other well-known methods for the set of test functions proposed by Dixon and Szegö [6] (cf. Section 5). Numerical results for one-dimensional test problems of a one-dimensional version of the method described can be found in Jansson [12], [13]; more results for other problems are given in Jansson and Knüppel [14].

2. The Method

In this section we present the minimization method in detail. Roughly speaking, it is a special branch and bound technique consisting of a repeated application of a bisection strategy in connection with a descent algorithm. The method consists of three algorithms. The first algorithm called "MINIMIZATION" implements a part of our branching strategy by calling the second algorithm "SUBDIVISION". The

latter determines a bisection process, discards subregions, improves starting points, and calls the third algorithm "SEARCH", which involves the descent algorithm.

We use the following notations. The bounds $\underline{X}, \overline{X}$ of a box $X := \{ x \in \mathbb{R}^n \mid \underline{X} \leq x \leq \overline{X} \}$ are called *lower* and *upper bound* of X. With $I(X)$ and $I(\mathbb{R}^n)$ we denote the set of all boxes contained in X and $\mathbb{R}^n$, resp. The *midpoint* of a box X is given by

$$m(X) := 0.5 \cdot (\underline{X} + \overline{X}), \tag{3}$$

the *width* of X is defined by

$$w(X) := \overline{X} - \underline{X}, \tag{4}$$

and the *relative width* is the vector $w_{\text{rel}}(X) := \left(w_{\text{rel}}(X_i)\right)_{i=1}^{n}$ with components

$$w_{\text{rel}}(X_i) := \begin{cases} w(X_i)/|m(X_i)| & \text{if} \quad 0 \notin X_i \\ w(X_i) & \text{otherwise} \end{cases} \tag{5}$$

where X_i denotes the i-th component of X. The *interval hull* of a set $Z \subseteq \mathbb{R}^n$ is defined by

$$\square(Z) := \bigcap \{ Y \in I(\mathbb{R}^n) \mid Z \subseteq Y \}. \tag{6}$$

The *distance* of two boxes $X, Y \in I(\mathbb{R}^n)$ is defined by (cf. [30], page 78)

$$d(X, Y) := \max\{ d_0(X, Y), d_0(Y, X) \} \tag{7}$$

where $d_0(x, Y) := \min_{y \in Y} \|x - y\|$, $d_0(X, Y) := \max_{x \in X} d_0(x, Y)$, and $\| \cdot \|$ denotes some norm. A sequence of boxes (X^k) with $X^k \in I(\mathbb{R}^n)$ *converges* to $x \in X$ if $\lim_{k \to \infty} d(X^k, x) = 0$. In this case we use the abbreviation $X^k \to x$.

For a function $f : X \to \mathbb{R}$ the *range* of f on a box $Y \subseteq X$ is denoted by $R(f(Y)) := \{ f(y) \mid y \in Y \}$. A function $F : I(X) \to I(\mathbb{R})$ is called an *inclusion function* of f on X, if

$$R(f(Y)) \subseteq F(Y) = [\underline{F}(Y), \overline{F}(Y)] \quad \text{for all} \quad Y \in I(X). \tag{8}$$

For the remainder of the paper we assume that an inclusion function F of f on X is given, and $n_{\text{it}}, n_{\text{d}} \in \mathbb{N} \setminus \{0\}$.

The two parameters $n_{\text{it}}, n_{\text{d}}$ of the method are responsible for the number of iterations and the number of bisections in each iteration step, respectively.

During the initialization (8.1), (8.2), (8.3) of MINIMIZATION the guaranteed lower and upper bounds $\underline{F}^*$ and $\overline{F}^*$ are set to $-\infty$ and ∞. List S holds the original box X in which we search for the global minimum, and $\underline{F}(X) := -\infty$. List A will contain the approximations to be calculated and is empty at the beginning.

The bounds $\underline{F}^*$, $\overline{F}^*$ and the lists S, A are global quantities w.r.t. all three procedures.

By passing through steps (8.4) to (8.11) at most n_{it} iterations are executed. In each iteration step $i = 1, \ldots, n_{\text{it}}$ SUBDIVISION is applied to all pairs contained in list S at the beginning of the iteration step. Later on, we will see that list S maintains the property

$$X^* \subseteq \bigcup \left\{ Y \mid (Y, \underline{F}(Y)) \in S \right\}. \tag{9}$$

Hence, in formula (8.10) $\underline{F}^*$ is updated and satisfies $\underline{F}^* \leq f^*$. Formula (8.11) provides an additional termination criterion. The algorithm terminates if the lower and upper bound of f^* are close enough, or if n_{it} iterations are executed.

procedure MINIMIZATION;
begin $Y^1 := X$, $\underline{F}^* := -\infty$, $\overline{F}^* := \infty$, $\underline{F}(Y^1) := \underline{F}^*$; (8.1)
initialize list $S := \left\{ (Y^1, \underline{F}(Y^1)) \right\}$, (8.2)
list of boxes containing all global minimum points;
initialize list $A := \emptyset$, (8.3)
list of approximations to be calculated;
for $i = 1, \ldots, n_{\text{it}}$ **do** (8.4)
begin
$S' := S$; (8.5)
$S := \emptyset$; (8.6)
for all pairs $(Y^j, \underline{F}(Y^j)) \in S'$ **do** (8.7)
begin
call SUBDIVISION for the pair $(Y^j, \underline{F}(Y^j))$; (8.8)
SUBDIVISION produces a list L that consists of pairs $(Z, \underline{F}(Z))$;
append list L calculated by SUBDIVISION (8.9)
at the end of list S;
end;
$\underline{F}^* := \text{Max}\left\{ \underline{F}^*, \text{Min}\{\underline{F}(Y) \mid (Y, \underline{F}(Y)) \in S\} \right\}$; (8.10)
if $(\overline{F}^* - \underline{F}^*) \leq \varepsilon$ **then** STOP (8.11)
end;
end;

The heart of our minimization method is SUBDIVISION (cf. (9.1) ... (9.15)). It is assumed that (i) a pair $(Y, \underline{F}(Y))$ with $Y \subseteq X$, (ii) a guaranteed upper bound $\overline{F}^*$ of f^*, and (iii) a permutation vector $p : \{1, \ldots, n\} \to \{1, \ldots, n\}$ are given. The permutation vector p determines in which direction the boxes are bisected. In all examples discussed in Section 5 p is choosen such that $w(X_{p(i)}) \geq w(X_{p(j)})$ for $i < j$, i.e., first the box is bisected normal to the direction with the largest width, then normal to the direction with the second largest width and so on.

In (9.1) $k_{\max}$ determines the maximal number of bisections which are performed in SUBDIVISION. It can be seen that the maximal length of list W equals $k_{\max}$.

The number $k(Y)$ determines the next direction of bisecting (cf. (9.8), (9.9)). The initialization $k(Y) := 0$ in (9.2) says that the starting box Y is bisected normal to $p(1)$, since in this case $s := (k(Y) \bmod n) + 1 = 1$. Hence, the starting box Y will be bisected normal to the direction with largest width of Y. Then in (9.3) a "working list" W containing the pair $(Y, \underline{F}(Y))$ together with $k(Y)$ and list $L := \emptyset$ are initialized.

Our scheme excludes most "uninteresting" parts of the search domain and concentrates on its "promising" subsets, and this should be done as fast as possible. Our experience is that if a box Y is bisected with $Y = Y^1 \cup Y^2$ and $w(Y^1) = w(Y^2)$, then in many cases the part Y^i with the smaller lower bound $\underline{F}(Y^i)$ $(i = 1, 2)$ will contain the global minimizer of f on Y. Passing through steps (9.4) to (9.15) we see that the algorithms proceeds always bisecting with box Y^i with the smaller lower bound (cf. (9.11), (9.12)), while it enters the box with the greater lower bound at the end of the working list W (cf. (9.13)). Thus, $k_{\max}$ bisections are executed on the original box Y, and then SEARCH is called in (9.14). The advantages of proceeding in this way are:

- The starting point of the descent method is improved by reducing the width of Y with the above heuristic: choose always the box with the smaller lower bound.
- The guaranteed upper bound $\overline{F}^*$, which excludes the uninteresting parts of Y in (9.7), (9.13), and (9.15), is quickly updated in (10.7). Notice that in (9.13), (9.15) boxes are no longer taken into consideration if $\underline{F}(Y^j) > \overline{F}^*$.

After calling SEARCH, the remaining boxes on list W are bisected. Only the boxes $Y^{(j)}$ which may contain global minimizers (i.e. $\underline{F}(Y^i) \leq \overline{F}^*$) are entered in a "local solution" list L (cf. (9.15)). List L is given back to algorithm MINIMIZATION in (8.9).

procedure SUBDIVISION;
Given a pair $(Y, \underline{F}(Y))$ and a permutation $p : \{1, \ldots, n\} \to \{1, \ldots, n\}$.

begin
$k_{\max} := n \cdot n_{\mathrm{d}}$; (9.1)
$k(Y) := 0$; (9.2)
initialize list $W := \left\{ (Y, \underline{F}(Y), k(Y)) \right\}$ and list $L := \emptyset$; (9.3)
while $W \neq \emptyset$ **do** (9.4)
begin
remove last triple $(Y, \underline{F}(Y), k(Y))$ from W; (9.5)
for $k = (k(Y) + 1), \ldots, k_{\max}$ **do** (9.6)
begin
if $\underline{F}(Y) > \overline{F}^*$ **then exit for loop** (9.7)
$s := (k \bmod n) + 1$; (9.8)
bisect Y normal to direction $p(s)$ getting two boxes

Y^1, Y^2 with $Y^1 \cup Y^2 = Y$; (9.9)
calculate $\underline{F}(Y^1), \underline{F}(Y^2)$; (9.10)
if $\underline{F}(Y^1) > \underline{F}(Y^2)$ **then**
exchange the indices of $(Y^1, \underline{F}(Y^1)), (Y^2, \underline{F}(Y^2))$; (9.11)
$Y := Y^1$; $\underline{F}(Y) := \underline{F}(Y^1)$; (9.12)
if $k < k_{\max}$ **and** $\underline{F}(Y^2) \leq \overline{F}^*$ **then**
enter the triple $(Y^2, F(Y^2), k)$ at the end of W; (9.13)
if $k = k_{\max}$ **and** $L = \emptyset$ **and** $\underline{F}(Y) \leq \overline{F}^*$ **then**
call SEARCH for $(Y, \underline{F}(Y))$; (9.14)
for $j = 1, 2$ **do**
if $k = k_{\max}$ **and** $\underline{F}(Y^j) \leq \overline{F}^*$ **then**
append $(Y^j, \underline{F}(Y^j))$ at the end of list L; (9.15)
end;
end;
end;

A key problem is deciding of when to call the descent algorithm. Obviously, if each call of SUBDIVISION would imply a call of the descent algorithm the computational costs may grow dramatically. Ideally, a descent method should be called, only if a global minimum point can indeed be calculated. This is the task of SEARCH. By (10.1), (10.2), we first calculate an approximation of $f_s := f(\text{mid}(Y))$, and then call the descent algorithm in (10.4), if (10.3a) or (10.3b) is satisfied.

The box $H(\tilde{x})$ is computed in (10.8) by means of an expansion around a local or global minimizer. Notice that for $\delta := 0$ the descent algorithm is called only if (10.3a) is fulfilled; in this case the descent method will surely improve $\overline{F}^*$ by (10.7). In our experience, using only (10.3a) has the disadvantage that many bisections have to be performed in situations where the first call of the descent algorithm delivers only a local minimum. This is, because the next call of the descent algorithm needs the midpoint of a box possessing a better value of the objective function than the previously calculated local minimum. Therefore, for the purpose of acceleration we additionally use condition (10.3b) with $\delta > 0$. Notice that for boxes Y inside of the expansion boxes $H(\tilde{x})$ the descent algorithm will only be called if condition (10.3a) is satisfied. The incorporation of expansion boxes are necessary for our method; otherwise, by condition (10.3b) the descent method would be called too many times, if the function f is very flat locally.

We emphasize that, in almost all of our test examples, the number of calls of the descent algorithm is equal to the number of global minimum points or at most threefold this number.

In (10.6), on a computer it is important to calculate the value $\overline{F}(\tilde{x})$ using the upper bound $\overline{F}$ of our inclusion function F with proper rounding. Then we know undoubtedly that $\overline{F}(\tilde{x})$ is a guaranteed upper bound of $f(\tilde{x})$ and therefore we cannot loose any global minimum points due to rounding errors. All other calculations in SEARCH are executed on a computer by using floating-point arithmetic.

procedure SEARCH;
Given a pair $(Y, \underline{F}(Y))$ and $\alpha, \beta, \gamma, \delta \geq 0$;
begin
$x_s := \text{mid}(Y)$; (10.1)
$f_s := f(x_s)$; (10.2)
if $f_s < \overline{F}^*$ **or** (10.3a)
if $Y \cap H(\tilde{x}) = \emptyset$ for each already calculated approximate
local or global minimum point stored in our
approximation list A and $f_s < \overline{F}^* + \delta \cdot |\overline{F}^*|$ **then** (10.3b)
begin
call a descent method with starting point x_s
calculating an approximation $\tilde{x}$; (10.4)
if $\tilde{x} \notin X$ **then** project $\tilde{x}$ orthogonal onto the bound of X, i.e.
if $\tilde{x}_i < \underline{X}_i$, then $\tilde{x}_i := \underline{X}_i$
if $\tilde{x}_i > \overline{X}_i$, then $\tilde{x}_i := \overline{X}_i$
if $\tilde{x}_i \in [\underline{X}_i, \overline{X}_i]$, then $\tilde{x}_i$ is not changed. (10.5)
$\tilde{f} := \overline{F}(\tilde{x})$; (10.6)
$\overline{F}^* := \min\{\tilde{f}, \overline{F}^*\}$; (10.7)
$H(\tilde{x}) := \tilde{x} \pm \alpha \cdot \text{Max}\{|\tilde{x} - x_s|, \beta|\tilde{x}|, \gamma\}$; (10.8)
append the triple $(\tilde{x}, \tilde{f}, H(\tilde{x}))$ at the end of list A; (10.9)
end;
end;

REMARKS.

1. In SUBDIVISION, step (9.11) gives a rule which says that the algorithm proceeds with the box possessing the smaller lower bound. In the rare case where both bounds are equal the algorithm proceeds with the first box. This case can be improved by inserting the following step after (9.10):

if $\underline{F}(Y^1) = \underline{F}(Y^2)$ then bisect Y normal to direction $p(s)$ yielding two boxes Y^1, Y^2 with $Y^1 \cup Y^2 = Y$ such that $w(Y^1) = 0.49 \cdot w(Y)$; then calculate $\underline{F}(Y^1), \underline{F}(Y^2)$;

This rule serves to accelerate our method especially, if a global minimum is close to or on the common boundary of Y^1 and Y^2.

2. Usually we start with a given box X. But we can also begin with a set of boxes and/or a set of approximations and bounds $\underline{F}^*, \overline{F}^*$; we have only to change the initialization (8.1), (8.2), (8.3) in MINIMIZATION. Moreover, the method can be used in an *interactive way*. That is, if the precision is not good enough we can successively increase i by applying (8.5) to (8.11) to the lists S, A, and the updated bounds $\underline{F}^*, \overline{F}^*$. In Section 4, numerical results are given. These should be read in the form that n_{it} is successively increased with fixed n_{d}.

Also by increasing i, the parameter n_{d} might be changed in SUBDIVISION guided by the computed results (for example the length of list S of the previous

iteration steps). For many problems values of n_d between 2 and 4 are satisfactory. Higher values of n_d are suited, if the region of attraction is very small.

Similarily, the parameter α, β, γ, δ and ε may be changed interactively. Thus, we have a great flexibility in applying our method.

Nevertheless for our numerical experiments in Section 4 the same set of parameters has been choosen identical for **all** test problems, with values $\alpha := 0.2$, $\beta := 0.1, \gamma := 10^{-3}, \delta := 0.2, \varepsilon := 0$. This heuristic parameters are not optimized w.r.t. to the set of problems discussed here. For some problems we did obtain better results by changing some of the parameters. The setting $\varepsilon := 0$ disables the termination criterion (8.11) and, thus, allows us to show the behaviour of our method for increasing parameters n_{it}, n_d.

3. Except for example 1, our implementation of the method uses the algorithm of Brent [3](cf. Chapter 7) as the local optimization method. For the non-differentiable test problems, special descent algorithms may give better results. In example 1 we used an SQP method, because this problem can also be viewed as a differentiable constrained optimization problem.

4. Step (9.7) in SUBDIVISION is our mechanism for deleting subboxes. If more information about the problem is given, then additional criteria can be incorporated, for example, monotonicity or concavity tests. Furthermore, knowing that the function f is concave on X permits us to discard boxes Y contained in the interior of X because here X^* is on the boundary of X.

3. Convergence

In this section we discuss convergence properties of our method. To do this we need some additional notation. The calculated bounds $\underline{F}^*, \overline{F}^*$ depend mainly on the parameters $n_{it}, n_d \in \mathbb{N}$, which determine the computational costs. In each iteration step $i = 1, \ldots, n_{it}$ (cf. (8.4)) of MINIMIZATION the lower bound $\underline{F}^*$ is updated in (8.10) whereas $\overline{F}^*$ is updated in (10.7). Therefore, we use the notation $\underline{F}^*(i, n_d), \overline{F}^*(i, n_d)$ to indicate the dependency on the iteration steps i and on n_d. $S(i, n_d)$ denotes S after executing the i-th loop (8.4), whereas $A(i, n_d)$ denotes A, the list of calculated approximations after executing loop (8.4). Moreover, let $L(Y, n_d)$ denote the list of pairs $(Z^k, \underline{F}(Z^k))$ generated by calling SUBDIVISION for a pair $(Y, \underline{F}(Y))$, and define

$$U(i, n_d) := \bigcup \{ Y^j \mid (Y^j, \underline{F}(Y^j)) \in S(i, n_d) \} \tag{11.1}$$

$$V(Y, n_d) := \bigcup \{ Z^k \mid (Z^k, \underline{F}(Z^k)) \in L(Y, n_d) \}. \tag{11.2}$$

The following convergence results hold for all acceleration parameters α, β, γ, $\delta \geq 0$. To show the asymptotic convergence behaviour, we define $n_{it} := \infty$ and omit the termination criterion (step (8.11) in MINIMIZATION) by setting $\epsilon := 0$. Hence, we do not mention this parameters in the following theorems. Our first

theorem gives the convergence behaviour of SUBDIVISION in dependence of $n_d \in \mathbb{N}$.

THEOREM 1. *Let $Y \subseteq X$ be a box, $\overline{F}^* \geq f^*$, and $p : \{1, \ldots, n\} \to \{1, \ldots, n\}$ be a permutation. Then SUBDIVISION applied to Y satisfies the following conditions:*

1. $Y \cap X^* \subseteq V(Y, n_d)$.
2. $w(Z^k) = w(Y)/2^{n_d}$ *for all* Z^k *with* $(Z^k, \underline{F}(Z^k)) \in L(Y, n_d)$.
3. *If the inclusion function F has the property that for each $x \in X$ and for each sequence $Z^k \to x$, $Z^k \subseteq X$, it follows that*

$$\underline{F}(Z^k) \to f(x), \tag{12}$$

and if $Y \cap X^* \neq \emptyset$ then

$$\mathrm{Min}\{\, \underline{F}(Z^k) \mid (Z^k, \underline{F}(Z^k)) \in L(Y, n_d) \,\} \to f^* \text{ as } n_d \to \infty. \tag{13}$$

Proof.

1. In SUBDIVISION pairs $(Y, \underline{F}(Y))$ are deleted only in three cases (9.7), (9.13), and (9.15), where $\underline{F}(Y)$ and $\underline{F}(Y^j), j = 1, 2$ are greater then $\overline{F}^*$. Hence, only pairs $(Y^j, \underline{F}(Y^j))$ with $X^* \cap Y^j = \emptyset$ are lost.

2. By (9.15) we see that pairs are only entered in list $L = L(Y, n_d)$ if $k = k_{\max} := n \cdot n_d$. Because p is a permutation each of the n coordinates of the starting box Y is bisected n_d times, proving our second statement.

3. Because of 1. and $Y \cap X^* \neq \emptyset$ it follows that

$$\mathrm{Min}\{\, \underline{F}(Z^k) \mid (Z^k, \underline{F}(Z^k)) \in L(Y, n_d) \,\} \leq f^* \tag{14}$$

for each $n_d \in \mathbb{N}$. Assume that (13) is not valid. Then it exists $\varepsilon > 0$ such that for all $n_d \in \mathbb{N}$

$$\mathrm{Min}\{\, \underline{F}(Z^k) \mid (Z^k, \underline{F}(Z^k)) \in L(Y, n_d) \,\} < f^* - \varepsilon. \tag{15}$$

Let $Z(n_d)$ denote the box with the lower bound $\underline{F}(Z(n_d))$ which is equal to the left handside of inequality (14) for each $n_d \in \mathbb{N}$. Because X is bounded, the sequence $(Z(n_d))_{n_d \in \mathbb{N}}$ is bounded. Let z^* be an accumulation point of this sequence. Let $(Z^k(n_d))_{k \in \mathbb{N}}$ be a convergent subsequence. Then $Z^k(n_d) \to z^*$ as $k \to \infty$ and by (12) $\underline{F}(Z^k(n_d)) \to f(z^*) \leq f^* - \varepsilon$, thereby contradicting the assumption that f^* is the global minimum value. ∎

Especially, Theorem 1 shows that in SUBDIVISION no global minimum points are lost and convergence of the lower bounds to f^* is assured. Condition (12) is very natural and in many cases computing inclusion functions satisfying property (12) causes no problems even, if f is not explicitly given. For example, in Section 5 some eigenvalue problems are discussed where the inclusion function is given by a numerical algorithm.

Let us now turn to the behaviour of algorithm MINIMIZATION. Now we assume that $n_d \in \mathbb{N}$ is fixed for all calls of SUBDIVISION. The following convergence results can easily be extended to the interactive case where n_d might be changed in each iteration step.

THEOREM 2. *Algorithm MINIMIZATION satisfies the following conditions:*

1. $X^* \subseteq U(i, n_d)$.
2. $f^* \in [\underline{F}^*(i, n_d), \overline{F}^*(i, n_d)]$, *and list* $A(i, n_d)$ *contains an approximation* $\tilde{x}$ *such that* $f(\tilde{x}) \in [\underline{F}^*(i, n_d), \overline{F}^*(i, n_d)]$.
3. $\underline{F}^*(i, n_d)$ *is monotonically increasing for increasing* i.
4. $\overline{F}^*(i, n_d)$ *is monotonically decreasing for increasing* i.
5. *If* $(Y, \underline{F}(Y)) \in S(i, n_d)$ *then* $w(Y) = w(X)/2^{n_d \cdot i}$.

Proof. 1. and 2. follows immediately by Theorem 1, (8.9), i.e. pairs are only deleted if their intersection with X^* is empty and the other pairs are entered into list S, and noticing the update of $\overline{F}^*$ in (10.7). By (8.10) it follows that $\underline{F}^*(i, n_d)$ is monotonically increasing and by (10.7) $\overline{F}^*(i, n_d)$ is monotonically decreasing for increasing i. 5. follows by Theorem 1, using (8.5) through (8.9), and noticing that we bisect boxes in SUBDIVISION w.r.t. a permutation p. ■

THEOREM 3. *Let the inclusion function* F *satisfy property (12) and let* $n_{it} := \infty$, *then the following holds for algorithm MINIMIZATION:*

1. $\lim_{i \to \infty} \overline{F}^*(i, n_d) = \lim_{i \to \infty} \underline{F}^*(i, n_d) = f^*$.
2. $U(i+1, n_d) \subseteq U(i, n_d)$ *and* $\lim_{i \to \infty} U(i, n_d) = X^*$.
3. $X^* = \bigcap_{i=1}^{\infty} U(i, n_d)$.

Proof. 1. By Theorem 1 it follows that the width of the boxes in list S decrease by factor 2^{n_d} in each iteration step i. Hence, (12) and Theorem 2 yields $\lim \underline{F}^*(i, n_d) = f^*$. Because a descent method in SEARCH is always called if condition (10.3a) is satisfied, we obtain with Theorem 2, $\lim_{i \to \infty} \overline{F}^*(i, n_d) = f^*$.

2. $U(i+1, n_d) \subseteq U(i, n_d)$ and $\lim U(i, n_d) \subseteq X^*$ are trivial consequences of our method (notice that no pairs are deleted that contain a global minimum point). If $\hat{x} \in \lim_{i \to \infty} U(i, n_d)$ then there is a sequence $Y^i \subseteq U(i, n_d)$, $i \in \mathbb{N}$, containing $\hat{x}$ as an accumulation point. By Theorem 1 $w(Y^i) \to 0$ with $i \to \infty$. By property (12) $\underline{F}(Y^i) \to f(\hat{x})$ as $Y^i \to \hat{x}$. By 1. we obtain $f(\hat{x}) = f^*$. Hence, $\hat{x} \in X^*$.

3. is a trivial consequence of 2. ■

Until now we have considered convergence properties for $i \to \infty$ and $n_d \to \infty$. The following theorem shows that after a **finite** number of iteration steps the global

minimum value f^* and a global minimum point $x^* \in X$ is calculated, provided a weak additional assumption is satisfied.

ASSUMPTION $(*)$. We assume that f is a continuous function, and that the descent algorithm is locally convergent for all $x^* \in X^*$. That is, there is a neighborhood $N(x^*)$ such that the descent algorithm converges to x^* for each starting point $x \in N(x^*)$ calculating x^* exactly if started in $N(x^*)$.

This assumption is weak because almost all descent algorithms are locally convergent for a wide class of problems. Moreover, Newton-type methods show locally superlinear or quadratic convergence. Therefore these methods compute x^* very fast and in principle arbitrarily accurate, provided the starting point is in $N(x^*)$. Hence, from a theoretical point of view, we can assume that the calculated approximation $\tilde{x}$ is identical with the corresponding global minimum point, provided that the starting point is contained in $N(x^*)$.

THEOREM 4. *If the assumption* $(*)$ *is satisfied, then there is an* $i_0 \in \mathbb{N}$ *such that list* $A(i_0, n_d)$ *contains a global minimum point and* $\overline{F}^*(i_0, n_d) = f^*$.

Proof. Assume that $A(i, n_d)$ contains no global minimum point for all $i \in \mathbb{N}$. Then $\overline{F}^*(i_0, n_d) > f^*$ for all $i \in \mathbb{N}$. Let (x^k) denote the finite or infinite sequence of points calculated by SEARCH being contained in $A(i, n_d)$ as $i \to \infty$. We have to consider two cases:

Case 1. There exists an $\epsilon > 0$ such that $f(x^k) \geq f^* + \epsilon$ for all k.

Let $x^* \in X^*$. Then by Theorem 2 (5.) there exists an $i_1 \in \mathbb{N}$ and a box Y^{i_1} with $x^* \in Y^{i_1}$, $Y^{i_1} \subseteq N(x^*)$, $(Y^{i_1}, \underline{F}(Y^{i_1})) \in S(i_1, n_d)$.

Because f is continuous Theorem 2 (5.) assures the existence of a box Y^{i_2}, $i_2 \geq i_1$ with $x^* \in Y^{i_2}$, $Y^{i_2} \subseteq Y^{i_1}$, $(Y^{i_2}, \underline{F}(Y^{i_2})) \in S(j_2, n_d)$ and $f(\operatorname{mid}(Y^{i_2})) < f^* + \epsilon$.

Hence, condition (10.3a) is satisfied; by SEARCH, the descent algorithm is called in iteration step $i_0 := i_2 + 1$ with starting point $\operatorname{mid}(Y^{i_2}) \in N(x^*)$, yielding $x^* \in A(i_0, n_d)$. This is a contradiction to the assumption that $A(i, n_d)$ contains no global minimum point for all $i \in \mathbb{N}$.

Case 2. $f(x^k) > f^*$ for all k and $f(x^k) \to f^*$ as $k \to \infty$.

The sequence (x^k) is bounded by X. Hence, there exists an accumulation point $\bar{x} \in X$ with $x^{k_j} \to \bar{x}$. Because f is continous, $f(\bar{x}) = f^*$ and $\bar{x} \in X^*$. This contradicts our assumption that the descent algorithm converges to $\bar{x}$ for all starting points in $N(\bar{x})$. ∎

Notice that in Theorem 4 we do not need assumptions about the quality of the inclusion function used. This is because the descent method is called in SEARCH using only function evaluations at real points (10.3a) and (10.3b). A similar theorem is not proved for the methods described in [10], [30]. As can be seen, demonstrated

by many test problems, our method typically computes an approximation of a global minimum point very rapidly. That is $\overline{F}^*(i, n_d) = f^*$ for very small i, n_d, whereas the lower bound $\underline{F}^*(i, n_d)$ may be not close to f^*. This is because in many situations the set of starting points that yield a global minimum point by the descent algorithm is large. Moreover, SUBDIVISION usually improves starting points for global minimizers.

4. Numerical Results

In this section we present the numerical results obtained by applying our method to a set of example functions. Examples 1–3 are non-differentiable problems with background in control theory and system analysis. The other test problems are well-known differentiable examples, where the set of functions given in examples 6–9 are commonly used for the comparison of global optimization methods.

The following additional abbreviations are used:

n_M denotes the number of global minima found by the algorithm, where a dash means, that only a local minimum has been found,

n_L denotes the number of calls of the descent method (cf. (10.4)),

l_s denotes the maximal length of the lists S, L, A,

n_{rf}, n_{if} are the total number of real and inclusion function calls used,

t is the machine independent standard unit time. The unit for t is the time needed to perform 1000 calls of the Shekel Function No. 5 at (4,4,4,4). On a SUN SparcStation 1 one unit in standard time is 0.25 s.

The algorithms described in Section 3 are implemented by using PROFIL/BIAS [16, 17, 18], a C++ library for numerical purposes including interval arithmetic. This library is freely available for non-commerical use on most workstations and PCs.

We emphasize that for all following test examples the computed approximations $\tilde{x}$ and $f(\tilde{x})$ of the global minima agree with the global minimum x^* and f^* with in at least six decimal digits. In the following we display in our tables only $\overline{F}^*$ rounded to six decimal digits, since $\overline{F}^*$ also agrees with $f(\tilde{x})$ and f^* in at least six decimal digits. The bounds for X^* are not given. In most cases these bounds are rough compared to $\overline{F}^* - \underline{F}^*$.

In examples 1 and 4 to 22 the inclusion functions used are natural interval extensions (cf. [29]). We mention that in some cases better results can be obtained by using centered forms or interval slopes.

EXAMPLE 1: The goal of the first example is to find for a system a lower-order model which in the minimax sense gives the best approximation to a system's impulse response.

The example has been taken from Charalambous and Bandler [4], where an approximation for a fourth-order system using a second-order model is searched. The fourth-order system has the transfer function

$$G(s) = \frac{(s+4)}{(s+1)(s^2+4s+8)(s+5)}.$$

The second-order model's transfer function is

$$H(s) = \frac{x_3}{(s+x_1)^2 + x_2^2},$$

where x_1, x_2, x_3 are the parameters of the model with $x_1, x_3 \in [0,1]$ and $x_2 \in [0.1, 1]$.

The impulse responses for the system and the model are:

$$\begin{aligned} s(t) &= \frac{3}{20}e^{-t} + \frac{1}{52}e^{-5t} - \frac{1}{65}e^{-2t}(3\sin 2t + 11\cos 2t) \\ h(x,t) &= \frac{x_3}{x_2}e^{-x_1 t}\sin x_2 t. \end{aligned}$$

The impulse responses are compared at 51 equidistant time points t_i, $i = 0, \ldots, 50$ in the time from 0 to 10 s.

The goal is finding a set of the three parameters for the model such that the maximal error $f(x) := \max_i |s(t_i) - h(x, t_i)|$ is minimal.

The solution is $f^* = 0.00794706$ at $x_1^* = 0.684418$, $x_2^* = 0.954093$, and $x_3^* = 0.122864$. Plots of $s(t)$ and $h(x^*, t)$ are shown in Figure 1, where the solid line is the impulse response of the model, and the dotted line is the system's response.

The results obtained by using our method are displayed in the following table.

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	0.00794706	1	2	283	236	1286	82.200
3	2	0.00631710	0.00794706	1	2	283	264	3836	224.267
4	2	0.00756954	0.00794706	1	2	283	293	7214	410.800
2	3	0.00631710	0.00794706	1	2	744	332	6046	332.800
3	3	0.00776569	0.00794706	1	2	744	344	11052	611.533
4	3	0.00791567	0.00794706	1	2	744	358	16232	899.333
2	4	0.00756954	0.00794706	1	2	471	346	7348	410.200

EXAMPLE 2: In system analysis, a commonly occuring problem is to minimize the maximal real part of the eigenvalues of a matrix in order to get a maximal stable system. The systems discussed here consist of a matrix $M(x) \in \mathbb{R}^{m \times m}$

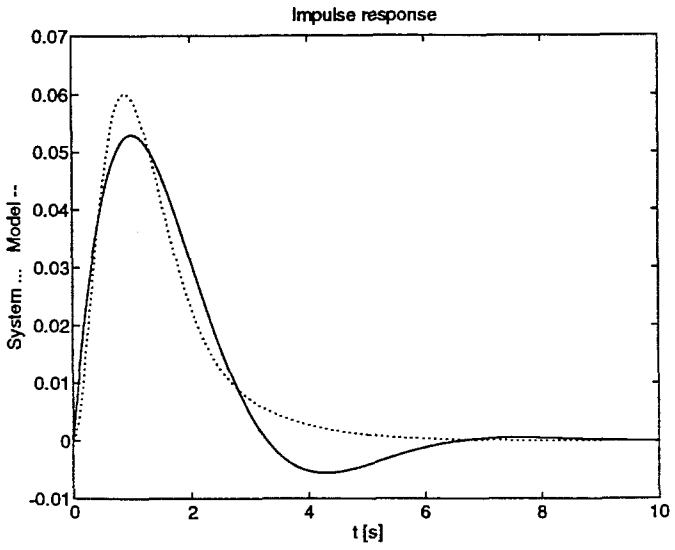

Fig. 1.

with parameters $x \in \mathbb{R}^2$. If $\lambda_i(x)$ are the eigenvalues of $M(x)$ and $\sigma(x) := \max_i \Re\{\lambda_i(x)\}$, then the goal is

$$\min_{x \in X} \sigma(x).$$

To get an inclusion function of $\sigma(x)$, the idea was to include eigenvalues by applying Gerschgorin's Theorem to $V^{-1}M(x)V$ where V is an approximation of the eigenvector matrix. We will omit a detailed description of how to compute this inclusion function but we mention that it is a variant of Lohner's method [20].

The first matrix we consider is

$$M(x) = \begin{pmatrix} d_1(x_1,x_2) & k\sin x_1 & k\sin x_2 & k\cos x_1 & k\cos x_2 \\ k\sin 2x_1 & d_2(x_1,x_2) & kx_1 & kx_2 & kx_1x_2 \\ k\sin 2x_2 & k(x_1+x_2) & d_3(x_1,x_2) & kx_1^2 & kx_2^2 \\ k\cos 2x_1 & k(x_1-x_2) & k(x_1+x_2)^2 & d_4(x_1,x_2) & k\sin x_1x_2 \\ k\cos 2x_2 & kx_1x_2^2 & k4x_2^2 & k\sin(x_1+x_2) & d_5(x_1,x_2) \end{pmatrix}$$

with

$$\begin{aligned} d_1(x_1,x_2) &= 17.5 - 2e^{-500((x_1+4)^2+(x_2+4)^2)} - \frac{x_1+x_2}{20} \\ d_2(x_1,x_2) &= 20 - \frac{x_1^2+x_2^2}{7} - (x_2+5)\cos\frac{\pi}{2}\left(x_1^2+x_2^2\right) \\ d_3(x_1,x_2) &= 20 - 6\cos 2\pi x_1 \\ d_4(x_1,x_2) &= 18 - \frac{x_1^4+x_2^4}{128} + \frac{1}{2}\cos 6\pi x_1x_2 \end{aligned}$$

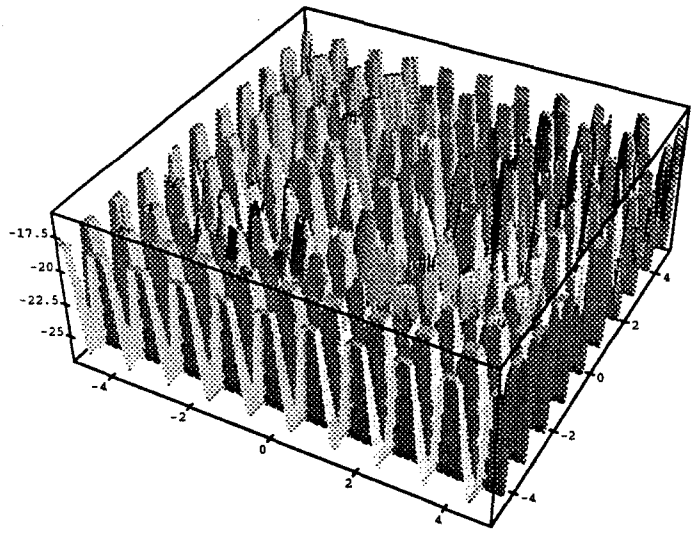

Fig. 2.

$$\begin{aligned} d_5(x_1, x_2) &= 20 - 6\cos 2\pi x_2 \\ k &= 10^{-3} \\ x_1, x_2 &\in [-5, 5] \end{aligned}$$

As it can be seen in the plot of $-\sigma(x)$ (we turned it upside down to let the global minimum be visible as global maximum), the function $\sigma(x)$ contains lots of local minima and maxima. The unique global minimum is

$$f^* = 15.9, \quad x^* = (-3.99997, -3.99997)$$

Applying our method, we obtain the results

n_{it}	n_{d}	$\underline{F}^*$	$\overline{F}^*$	n_{M}	n_{L}	l_{s}	n_{rf}	n_{if}	t
2	2	15.8686	17.1890	–	2	44	815	114	144.3
2	4	15.8974	15.9000	1	1	1	133	33	21.5
3	4	15.8999	15.9000	1	1	1	134	49	28.5

EXAMPLE 3: With the matrix

$$M(x) = \begin{pmatrix} d & k\sin x_1 & k\sin x_2 \\ k\sin x_1 & 15\sin\pi(x_2 + 3.75) - 7 & kx_2^2/10 \\ k\sin x_2 & kx_2^2/10 & 15\sin\pi(x_2 + 3.75) - 7 \end{pmatrix}$$

and

$$d = x_1^2/2 - 37e^{-80p} + \sin\pi(5p - 0.25) + 2$$

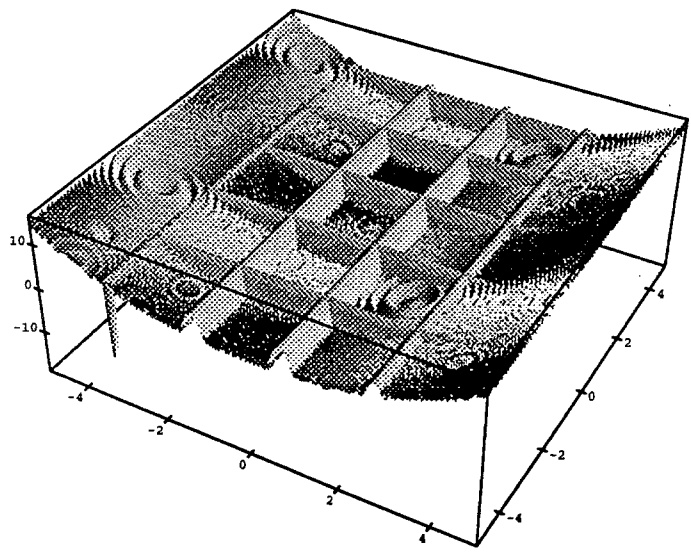

Fig. 3.

$$\begin{aligned} p &= (x_1+4)^2+(x_2+4)^2 \\ k &= 10^{-3} \\ x_1, x_2 &\in [-5,5] \end{aligned}$$

the plot of $\sigma(x)$ is given in Figure 3, with the global minimum

$$f^* = -18.8718, \quad x^* = (-4.04074, -4.04074)$$

Applying our method, we obtain the following results

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	−27.3070	−18.8718	1	2	9	660	36	13.9
2	4	−19.8697	−18.8718	1	1	1	294	33	11.1
3	4	−18.9589	−18.8718	1	1	4	295	75	26.3
4	4	−18.8852	−18.8718	1	1	11	296	143	41.9

EXAMPLE 4: Photoelectron Spectroscopy Problem. This example is taken from Moore, Hansen, and Leclerc [24]: in the field of chemistry, a very common problem is to reconstruct a curve that is given by n points (x_i, y_i), $i = 1, \ldots, n$. Normally, the curve is a sum of peaks and the chemist desires to resolve the shape and the position of the individual peaks. In [24] the curve is given as a sum of two gaussian

peaks:

$$
\begin{aligned}
x_i &= 4 + (i+1)/10,\ i = 1, 2, \ldots, 81 \\
y_i &= a_1 \cdot \exp\left(-\left((x_i - u_1)/s_1\right)^2\right) + a_2 \cdot \exp\left(-\left((x_i - u_2)/s_2\right)^2\right) \\
a_1 &= 130.89 \qquad a_2 = 52.6 \\
u_1 &= 6.73 \qquad u_2 = 9.342 \\
s_1 &= 1.2 \qquad s_2 = 0.97
\end{aligned}
$$

The goal is to recover the six parameters $a_1, a_2, u_1, u_2, s_1, s_2$ of the curve function

$$p(x, a_1, a_2, u_1, u_2, s_1, s_2) = \sum_{j=1}^{2} a_j \cdot \exp\left(-\left((x - u_j)/s_j\right)^2\right)$$

such that the error

$$\sum_{i=1}^{81} \left(p(x_i, a_1, a_2, u_1, u_2, s_1, s_2) - y_i\right)^2$$

is minimized.

The range of the six parameters is defined as follows:

$$
\begin{aligned}
a_1 &\in [130, 135] \qquad a_2 \in [50, 55] \\
u_1 &\in [6, 8] \qquad u_2 \in [8, 10] \\
s_1 &\in [1, 2] \qquad s_2 \in [0.5, 1]
\end{aligned}
$$

The method described in [24] uses interval derivatives in an extensive way and obtains the following guaranteed bounds for the global minimum value and the global minimum point:

$$
\begin{aligned}
a_1 &= [130.889999624668920, 130.890000237423440] \\
a_2 &= [52.5999994426222910, 52.6000003353821410] \\
u_1 &= [6.72999999580056230, 6.73000000523584680] \\
u_2 &= [9.34199999170696670, 9.34200000792551850] \\
s_1 &= [1.19999999502502950, 1.20000000672384770] \\
s_2 &= [0.96999998507893725, 0.97000001469388031] \\
f &= [6.3015390640\ldots \cdot 10^{-13}, 9.9696829305\ldots \cdot 10^{-11}]
\end{aligned}
$$

The number of correct digits for the global minimum point is 7 . . . 9. The computation time needed on a SUN SparcStation 1 is reported as 109240 s.

If we apply our method, we get approximations with 12 . . . 15 correct digits

$$
\begin{aligned}
\tilde{a}_1 &= 130.8900000000426 \\
\tilde{a}_2 &= 52.59999999998969 \\
\tilde{u}_1 &= 6.730000000000008 \\
\tilde{u}_2 &= 9.342000000000636 \\
\tilde{s}_1 &= 1.199999999999803 \\
\tilde{s}_2 &= 0.9700000000003587
\end{aligned}
$$

and the following further results:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
1	1	0	$2.72132 \cdot 10^{-20}$	1	1	8	256	53	12.4

This means, the computation time is about 3.1 s on a SUN SparcStation 1.

EXAMPLE 5: Griewank function [35] ($n = 2, 10, 50$).

$$f_G(x) = \sum_{i=1}^{n} x_i^2/d - \prod_{i=1}^{n} \cos\frac{x_i}{\sqrt{i}} + 1$$

$$-100 \le x_1, x_2 \le 100, \quad d = 200, \qquad \text{for } n = 2$$

$$-600 \le x_i \le 600, \quad d = 4000, \qquad \text{for } n = 10, 50$$

$$f^* = 0, \quad x^* = (0, \ldots, 0)$$

For $n = 2$ we obtain the following results:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	0.781741	–	1	8	123	65	0.467
3	2	0	$9.99201 \cdot 10^{-16}$	1	2	8	201	74	0.800
4	2	0	$9.99201 \cdot 10^{-16}$	1	2	8	202	82	0.867
2	3	0	$9.99201 \cdot 10^{-16}$	1	2	4	186	58	0.733
3	3	0	$9.99201 \cdot 10^{-16}$	1	2	4	187	70	0.800
4	3	0	$9.99201 \cdot 10^{-16}$	1	2	4	188	82	0.800
2	4	0	$9.99201 \cdot 10^{-16}$	1	2	4	154	78	0.733

Using $n = 10$ and applying our method leads to

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	8	0	$1.31006 \cdot 10^{-14}$	1	1	1	135	341	2.667
2	10	0	$1.80300 \cdot 10^{-13}$	1	1	1	417	421	4.600
2	12	0	$3.25184 \cdot 10^{-13}$	1	1	1	400	501	5.133

Only for a few real methods results are known for this function. In Törn and Žilinskas [35] the results of two methods are given:

Method	n_M	n_{rf}	t
Griewank (1981)	–	6600	–
Snyman, Fatti (1987)	1	23399	90

Further for n to 50, we get

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
1	10	0	$1.14087 \cdot 10^{-12}$	1	1	1	7644	1101	110.333
2	10	0	$1.14087 \cdot 10^{-12}$	1	1	1	7645	2101	140.333
1	15	0	$2.25375 \cdot 10^{-14}$	1	1	1	743	1601	48.067

EXAMPLE 6: Branin function [35].

$$f_{BR}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$$

$$-5 \le x_1 \le 10, \quad 0 \le x_2 \le 15$$

$$f^* = 0.397887, \quad x^* = \begin{cases} (-3.14159, 12.27500) \\ (\ 3.14159,\ \ 2.27500) \\ (\ 9.42478,\ \ 2.47500) \end{cases}$$

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0.397887	0.397887	1	1	7	127	79	0.600
3	2	0.397887	0.397887	3	3	9	271	139	1.267
4	2	0.397887	0.397887	3	3	9	276	201	1.467
2	3	0.397887	0.397887	3	3	10	247	143	1.200
3	3	0.397887	0.397887	3	3	10	251	231	1.400
4	3	0.397887	0.397887	3	3	10	254	309	1.600
2	4	0.397887	0.397887	3	3	9	233	209	1.267

EXAMPLE 7: Goldstein–Price function [35].

$$\begin{aligned} f_{GP}(x) = {} & [1 + (x_1 + x_2 + 1)^2 \\ & (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times \\ & [30 + (2x_1 - 3x_2)^2 \\ & (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)] \end{aligned}$$

$$-2 \le x_1, x_2 \le 2$$

$$f^* = 3, \quad x^* = (0, -1)$$

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
1	1	3.00000	3.00000	1	1	2	99	9	0.333
2	2	3.00000	3.00000	1	1	2	121	39	0.533
3	2	3.00000	3.00000	1	1	2	123	55	0.467
4	2	3.00000	3.00000	1	1	3	124	79	0.667
2	3	3.00000	3.00000	1	1	2	126	59	0.600
3	3	3.00000	3.00000	1	1	3	127	93	0.667
4	3	3.00000	3.00000	1	1	3	128	125	0.667
2	4	3.00000	3.00000	1	1	3	108	87	0.533

EXAMPLE 8: Shekel functions [35].

$$f_{Sm}(x) = -\sum_{i=1}^{m} \frac{1}{(x - a_i)(x - a_i)^T + c_i}$$

$$0 \le x_i \le 10, \quad i = 1, \ldots, 4$$

The coefficients are:

i	a_i	c_i
1	(4.0, 4.0, 4.0, 4.0)	0.1
2	(1.0, 1.0, 1.0, 1.0)	0.2
3	(8.0, 8.0, 8.0, 8.0)	0.2
4	(6.0, 6.0, 6.0, 6.0)	0.4
5	(3.0, 7.0, 3.0, 7.0)	0.4
6	(2.0, 9.0, 2.0, 9.0)	0.6
7	(5.0, 5.0, 3.0, 3.0)	0.3
8	(8.0, 1.0, 8.0, 1.0)	0.7
9	(6.0, 2.0, 6.0, 2.0)	0.5
10	(7.0, 3.6, 7.0, 3.6)	0.5

For $m = 5$ the minimum is:

$$f^* = -10.1532, \quad x^* = (4.00004, 4.00013, 4.00004, 4.00013)$$

We obtain the following results:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	−10.2083	−10.1532	1	1	1	165	33	0.733
3	2	−10.1622	−10.1532	1	1	1	166	49	0.800
4	2	−10.1559	−10.1532	1	1	1	167	65	0.867
2	3	−10.1622	−10.1532	1	1	1	151	49	0.733
3	3	−10.1540	−10.1532	1	1	1	152	85	0.867
4	3	−10.1534	−10.1532	1	1	7	153	205	1.267
2	4	−10.1559	−10.1532	1	1	1	97	65	0.600

For $m = 7$ the minimum is:

$$f^* = -10.4029, \quad x^* = (4.00057, 4.00069, 3.99949, 3.99961)$$

Applying our method, we get:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	−10.6818	−10.4029	1	1	1	193	37	0.867
3	2	−10.4501	−10.4029	1	1	1	194	53	1.000
4	2	−10.4144	−10.4029	1	1	1	195	85	1.133
2	3	−10.4501	−10.4029	1	1	1	138	53	0.733
3	3	−10.4100	−10.4029	1	1	13	139	115	1.067
4	3	−10.4040	−10.4029	1	1	70	142	691	3.933
2	4	−10.4144	−10.4029	1	1	1	86	85	0.733

For $m = 10$ the minimum is:

$$f^* = -10.5364, \quad x^* = (4.00075, 4.00059, 3.99966, 3.99951)$$

Again applying our method, we get:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	−10.8593	−10.5364	1	1	1	171	39	0.933
3	2	−10.5918	−10.5364	1	1	1	172	55	1.067
4	2	−10.5501	−10.5364	1	1	1	173	93	1.267
2	3	−10.5918	−10.5364	1	1	1	144	55	0.800
3	3	−10.5447	−10.5364	1	1	15	145	123	1.267
4	3	−10.5376	−10.5364	1	1	92	148	803	5.667
2	4	−10.5501	−10.5364	1	1	1	86	93	0.933

EXAMPLE 9: Hartman functions ($n = 3, 6$) [35].

$$f_{Hm}(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{n} \alpha_{ij}(x_j - p_{ij})^2\right]$$

$$0 \le x_i \le 1, \quad i = 1, \ldots, n$$

Choosing $n = 3$, the coefficients are:

i	α_i	c_i	p_i
1	(3.0, 10.0, 30.0)	1.0	(0.36890, 0.11700, 0.26730)
2	(0.1, 10.0, 35.0)	1.2	(0.46990, 0.43870, 0.74700)
3	(3.0, 10.0, 30.0)	3.0	(0.10910, 0.87320, 0.55470)
4	(0.1, 10.0, 35.0)	3.2	(0.03815, 0.57430, 0.88280)

and the global minimum is

$$f^* = -3.86278, \quad x^* = (0.114614, 0.555649, 0.852547)$$

Applying our method leads to the following results:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
1	2	−5.49740	−3.08976	1	1	21	117	67	1.000
2	2	−4.32854	−3.86278	1	5	107	571	689	4.733
3	2	−3.98711	−3.86278	1	5	867	594	5375	23.067
4	2	−3.89500	−3.86278	1	5	7902	831	47571	190.200
2	3	−3.98711	−3.86278	1	4	867	638	5368	23.600
2	4	−3.89500	−3.86278	1	4	7815	702	47564	231.867

Choosing $n = 6$, the coefficients are:

i	α_i	c_i
1	(10.00, 3.00, 17.00, 3.50, 1.70, 8.00)	1.0
2	(0.05, 10.00, 17.00, 0.10, 8.00, 14.00)	1.2
3	(3.00, 3.50, 1.70, 10.00, 17.00, 8.00)	3.0
4	(17.00, 8.00, 0.05, 10.00, 0.10, 14.00)	3.2

i	p_i
1	(0.1312, 0.1696, 0.5569, 0.0124, 0.8283, 0.5886)
2	(0.2329, 0.4135, 0.8307, 0.3736, 0.1004, 0.9991)
3	(0.2348, 0.1451, 0.3522, 0.2883, 0.3047, 0.6650)
4	(0.4047, 0.8828, 0.8732, 0.5743, 0.1091, 0.0381)

The global minimum is

$$f^* = -3.32237,$$
$$x^* = (0.201690, 0.150011, 0.476874, 0.275332, 0.311652, 0.657300)$$

If we apply our method, we get

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
1	2	−4.14692	−3.32237	1	2	65	464	780	6.267
2	2	−3.51182	−3.32237	1	2	685	479	6744	42.067

EXAMPLE 10: Levy No. 3 [36]

$$f_{whs4}(x) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right) \cdot \left(\sum_{j=1}^{5} j\cos((j+1)x_2 + j)\right)$$

$$-10 \le x_i \le 10, \quad i = 1, 2$$

$$f^* = -176.542, \quad x^* = \begin{cases} (4.97648, 4.85806) \\ (4.97648, -1.42513) \\ (4.97648, -7.70831) \\ (-1.30671, 4.85806) \\ (-1.30671, -1.42513) \\ (-1.30671, -7.70831) \\ (-7.58989, 4.85806) \\ (-7.58989, -1.42513) \\ (-7.58989, -7.70831) \end{cases}$$

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	−215.521	−176.542	1	1	136	111	459	2.733
3	2	−208.402	−176.542	5	5	136	485	1195	8.733
4	2	−191.294	−176.542	9	9	136	822	1521	12.400
2	3	−208.402	−176.542	5	7	79	651	1345	9.733
3	3	−184.481	−176.542	9	11	131	952	2029	15.533
4	3	−177.638	−176.542	9	11	986	994	6827	71.800
2	4	−191.294	−176.542	8	10	269	834	1658	13.333

EXAMPLE 11: Levy No. 5 [36]

$$f_{whs5}(x) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right) \cdot \left(\sum_{j=1}^{5} j\cos((j+1)x_2 + j)\right) + (x_1 + 1.42513)^2 + (x_2 + 0.80032)^2$$

$$-10 \le x_i \le 10, \quad i = 1, 2$$

$$f^* = -176.138, \quad x^* = (-1.30685, -1.42485)$$

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	−217.902	−8.4488	–	1	250	133	507	3.467
3	2	−205.369	−176.138	1	2	250	218	632	4.733
4	2	−189.796	−176.138	1	2	250	220	666	4.933
2	3	−205.369	−176.138	1	1	30	89	309	2.000
3	3	−183.934	−176.138	1	1	30	91	383	2.467
4	3	−177.200	−176.138	1	1	117	96	915	9.300
2	4	−189.796	−176.138	1	2	252	199	664	5.067

EXAMPLE 12: Levy No. 8–12 ($n = 3, 4, 5, 8, 10$) [36]

$$f_{levy1}(x) = \sin^2 \pi y_1 + \sum_{i=1}^{n-1} (y_i - 1)^2 (1 + 10\sin^2 \pi y_{i+1}) + (y_n - 1)^2$$

$$\text{with } y_i = 1 + (x_i - 1)/4$$

$$-10 \le x_i \le 10, \quad i = 1, \ldots, n$$

$$f^* = 0, \quad x^* = (1, \ldots, 1)$$

For $n = 3$ the results are:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	$8.38942 \cdot 10^{-30}$	1	1	1	106	25	0.533
3	2	0	$8.38942 \cdot 10^{-30}$	1	1	1	107	37	0.533
4	2	0	$8.38942 \cdot 10^{-30}$	1	1	1	108	49	0.600
2	3	0	$1.49966 \cdot 10^{-32}$	1	1	1	84	37	0.400
3	3	0	$1.49966 \cdot 10^{-32}$	1	1	1	85	55	0.467
2	4	0	$3.22519 \cdot 10^{-31}$	1	1	1	73	49	0.400

For $n = 4$ we applied our method with the following results:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	$5.98650 \cdot 10^{-31}$	1	1	1	114	33	0.467
3	2	0	$5.98650 \cdot 10^{-31}$	1	1	1	115	49	0.600
4	2	0	$5.98650 \cdot 10^{-31}$	1	1	1	116	65	0.733
2	3	0	$6.66442 \cdot 10^{-31}$	1	1	1	101	49	0.533
3	3	0	$6.66442 \cdot 10^{-31}$	1	1	1	102	73	0.733
2	4	0	$1.42632 \cdot 10^{-30}$	1	1	1	87	65	0.600

Increasing n to 5 leads to:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	$6.54453 \cdot 10^{-18}$	1	1	1	156	41	0.800
3	2	0	$6.54453 \cdot 10^{-18}$	1	1	1	157	61	0.867
4	2	0	$6.54453 \cdot 10^{-18}$	1	1	1	158	81	1.000
2	3	0	$7.58057 \cdot 10^{-25}$	1	1	1	123	61	0.733
3	3	0	$7.58057 \cdot 10^{-25}$	1	1	1	124	91	0.867
2	4	0	$3.52595 \cdot 10^{-28}$	1	1	1	118	81	0.800

Further increasing n to 8 yields:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	$7.25593 \cdot 10^{-19}$	1	1	1	244	65	1.400
3	2	0	$7.25593 \cdot 10^{-19}$	1	1	1	245	97	1.800
4	2	0	$7.25593 \cdot 10^{-19}$	1	1	1	246	129	2.000
2	3	0	$9.70333 \cdot 10^{-21}$	1	1	1	255	97	1.667
3	3	0	$9.70333 \cdot 10^{-21}$	1	1	1	256	145	2.000
2	4	0	$1.81866 \cdot 10^{-25}$	1	1	1	241	129	1.800

With $n = 10$, the results are:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	$3.41336 \cdot 10^{-16}$	1	1	1	379	81	2.200
3	2	0	$3.41336 \cdot 10^{-16}$	1	1	1	380	121	2.600
4	2	0	$3.41336 \cdot 10^{-16}$	1	1	1	381	161	2.933
2	3	0	$1.49337 \cdot 10^{-12}$	1	1	1	146	121	1.667
3	3	0	$1.49337 \cdot 10^{-12}$	1	1	1	147	181	2.133
2	4	0	$4.93934 \cdot 10^{-21}$	1	1	1	345	161	2.933

To show the behaviour for higher dimensions, we enlarge n to 50. This problem is not contained in the test set [36]. The results are:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	$1.97959 \cdot 10^{-13}$	1	1	1	6075	401	103.700
3	2	0	$1.97959 \cdot 10^{-13}$	1	1	1	6076	601	116.300
4	2	0	$1.97959 \cdot 10^{-13}$	1	1	1	6077	801	128.500
2	3	0	$1.04201 \cdot 10^{-12}$	1	1	1	666	601	41.500
3	3	0	$1.04201 \cdot 10^{-12}$	1	1	1	667	901	58.700
4	3	0	$1.04201 \cdot 10^{-12}$	1	1	1	668	1201	76.100
2	4	0	$2.92226 \cdot 10^{-18}$	1	1	1	5565	801	120.300

EXAMPLE 13: Levy No. 13–18 ($n = 2, 3, 4, 5, 7$) [36]

$$f_{levy2}(x) = \sin^2 3\pi x_1 + \sum_{i=1}^{n-1} (x_i - 1)^2 \left(1 + \sin^2 3\pi x_{i+1}\right) + (x_n - 1)^2 \left(1 + \sin^2 2\pi x_n\right)$$

$$-10 \le x_i \le 10, \quad i = 1, \ldots, n \quad \text{for } n \le 4$$

$$-5 \le x_i \le 5, \quad i = 1, \ldots, n \quad \text{for } n > 4$$

$$f^* = 0, \quad x^* = (1, \ldots, 1)$$

For $n = 2$ the results are:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	0.439489	–	1	4	113	19	0.467
3	2	0	0.439489	–	1	22	117	85	0.733
4	2	0	$3.60551 \cdot 10^{-30}$	1	2	22	184	94	1.067
2	3	0	0.109874	–	1	8	133	47	0.533
3	3	0	$3.60551 \cdot 10^{-30}$	1	2	8	190	60	0.800
2	4	0	$3.60551 \cdot 10^{-30}$	1	2	3	154	36	0.667

With $n = 3$ we yield:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	0.210238	–	2	8	343	34	1.267
3	2	0	0.210238	–	2	51	351	256	2.133
4	2	0	$9.9801 \cdot 10^{-27}$	1	3	51	421	269	2.400
2	3	0	0.109874	–	1	20	159	101	0.800
3	3	0	$5.79447 \cdot 10^{-29}$	1	2	20	205	120	1.133
2	4	0	$1.22359 \cdot 10^{-26}$	1	2	8	203	58	0.867

Applying our method for $n = 4$, we get:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	0.308268	–	2	16	396	56	1.600
3	2	0	0.308268	–	2	209	412	1030	5.667
4	2	0	$5.63553 \cdot 10^{-27}$	1	3	209	502	1047	6.267
2	3	0	0.109874	–	1	48	181	201	1.467
3	3	0	$1.48838 \cdot 10^{-15}$	1	2	48	249	226	1.867
2	4	0	$5.63855 \cdot 10^{-27}$	1	2	16	256	88	1.267

Increasing n to 5, we yield:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	0.210238	–	1	51	271	441	2.733
3	2	0	$5.59851 \cdot 10^{-14}$	1	2	51	364	462	3.200
4	2	0	$5.59851 \cdot 10^{-14}$	1	2	51	365	482	3.333
2	3	0	$3.05564 \cdot 10^{-14}$	1	2	32	283	114	1.533
3	3	0	$3.05564 \cdot 10^{-14}$	1	2	32	284	144	1.667
2	4	0	$4.15295 \cdot 10^{-28}$	1	1	1	120	81	0.800

Further increasing n to 7 leads to:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	0.109874	–	1	8	395	197	2.467
3	2	0	$1.78379 \cdot 10^{-18}$	1	2	8	568	226	3.400
4	2	0	$1.78379 \cdot 10^{-18}$	1	2	8	569	254	3.533
2	3	0	$1.30769 \cdot 10^{-17}$	1	2	99	588	310	3.933
3	3	0	$1.30769 \cdot 10^{-17}$	1	2	99	589	352	4.200
2	4	0	$3.26942 \cdot 10^{-23}$	1	1	1	206	113	1.400

As in the previous example, we enlarge the dimension to $n = 50$, which leads to:

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	0.109874	–	1	51	7133	7851	356.000
3	2	0	$9.35854 \cdot 10^{-15}$	1	2	51	7798	8052	385.300
4	2	0	$9.35854 \cdot 10^{-15}$	1	2	51	7799	8252	390.800
2	3	0	$7.83971 \cdot 10^{-15}$	1	2	51	9176	3052	207.300
3	3	0	$7.83971 \cdot 10^{-15}$	1	2	51	9177	3352	219.000
4	3	0	$7.83971 \cdot 10^{-15}$	1	2	51	9178	3652	236.900
2	4	0	$7.98569 \cdot 10^{-16}$	1	1	1	5786	801	105.100

EXAMPLE 14: Schwefel No. 1.2 [36]

$$f_{S1.2}(x) = \sum_{i=1}^{4} \left(\sum_{j=1}^{i} x_j \right)^2$$

$$-5 \le x_i \le 10, \quad i = 1, \ldots, 4$$

$$f^* = 0, \quad x^* = (1, 1, 1, 1)$$

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	$1.69744 \cdot 10^{-52}$	1	1	28	128	391	1.267
3	2	0	$1.69744 \cdot 10^{-52}$	1	1	34	131	719	1.933
4	2	0	$1.69744 \cdot 10^{-52}$	1	1	34	134	899	2.267
2	3	0	$6.41699 \cdot 10^{-47}$	1	1	34	118	727	1.867
3	3	0	$6.41699 \cdot 10^{-47}$	1	1	34	120	1001	2.400

EXAMPLE 15: Beale [36]

$$f_{BE}(x) = \left(1.5 - x_1 + x_1 x_2\right)^2 + \left(2.25 - x_1 + x_1 x_2^2\right)^2 + \left(2.625 - x_1 + x_1 x_2^3\right)^2$$

$$-4.5 \le x_i \le 4.5, \quad i = 1, 2$$

$$f^* = 0, \quad x^* = (3, 0.5)$$

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	$3.23393 \cdot 10^{-18}$	1	2	14	351	102	1.400
3	2	0	$2.39976 \cdot 10^{-18}$	2	3	17	467	253	2.133
4	2	0	$2.39976 \cdot 10^{-18}$	2	3	17	469	383	2.467
2	3	0	$4.76286 \cdot 10^{-23}$	1	2	20	260	266	1.400
3	3	0	$4.76286 \cdot 10^{-23}$	1	2	20	262	442	1.733
2	4	0	$4.02320 \cdot 10^{-16}$	1	2	26	419	422	2.200

EXAMPLE 16: Schwefel No. 3.1 [36]

$$f_{S3.1}(x) = \sum_{i=1}^{3} \left[\left(x_1 - x_i^2\right)^2 + (x_i - 1)^2\right]$$

$$-10 \le x_i \le 10, \quad i = 1, \ldots, 3$$

$$f^* = 0, \quad x^* = (1, 1, 1)$$

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	$5.96423 \cdot 10^{-22}$	1	1	1	102	25	0.467
3	2	0	$5.96423 \cdot 10^{-22}$	1	1	1	103	37	0.467
4	2	0	$5.96423 \cdot 10^{-22}$	1	1	1	104	49	0.467
2	3	0	$9.55520 \cdot 10^{-23}$	1	1	1	81	37	0.400
3	3	0	$9.55520 \cdot 10^{-23}$	1	1	1	82	55	0.467
2	4	0	$1.04758 \cdot 10^{-22}$	1	1	1	96	49	0.467

EXAMPLE 17: Booth [36]

$$f_{BO}(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

$$-10 \leq x_i \leq 10, \quad i = 1, 2$$

$$f^* = 0, \quad x^* = (1, 3)$$

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	$1.57772 \cdot 10^{-30}$	1	1	4	66	49	0.467
3	2	0	$1.57772 \cdot 10^{-30}$	1	1	4	68	69	0.400
4	2	0	$1.57772 \cdot 10^{-30}$	1	1	4	70	93	0.533
2	3	0	0	1	1	2	77	71	0.467
3	3	0	0	1	1	3	79	109	0.533
2	4	0	0	1	1	3	94	99	0.600

EXAMPLE 18: Kowalik [36]

$$f_K(x) = \sum_{i=1}^{11} \left(a_i - x_1 \frac{b_i^2 + b_i x_2}{b_i^2 + b_i x_3 + x_4} \right)^2$$

$$0 \leq x_i \leq 0.42, \quad i = 1, \ldots, 4$$

The coefficients are:

i	a_i	$1/b_i$
1	0.1957	0.25
2	0.1947	0.50
3	0.1735	1.00
4	0.1600	2.00
5	0.0844	4.00
6	0.0627	6.00
7	0.0456	8.00
8	0.0342	10.00
9	0.0323	12.00
10	0.0235	14.00
11	0.0246	16.00

$$f^* = 3.07486 \cdot 10^{-4}, \quad x^* = (0.192833, 0.190836, 0.123117, 0.135766)$$

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	1	0	$3.07486 \cdot 10^{-4}$	1	1	79	213	315	2.600
3	1	0	$3.07486 \cdot 10^{-4}$	1	1	361	285	1499	10.067

EXAMPLE 19: Powell [36]

$$f_{Pow}(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

$$-4 \leq x_i \leq 5, \quad i = 1, \ldots, 4$$

$$f^* = 0, \quad x^* = (0, 0, 0, 0)$$

The Hessian is singular at x^*.

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	$5.96299 \cdot 10^{-18}$	1	1	8	481	185	1.933
3	2	0	$5.96299 \cdot 10^{-18}$	1	1	8	482	283	2.267
4	2	0	$5.96299 \cdot 10^{-18}$	1	1	8	484	393	2.600
2	3	0	$4.63594 \cdot 10^{-17}$	1	1	4	537	285	2.533
3	3	0	$4.63594 \cdot 10^{-17}$	1	1	5	538	445	2.733
2	4	0	$3.23044 \cdot 10^{-20}$	1	1	7	570	403	2.867

Enlarging the domain to

$$-2 \cdot 10^6 \leq x_i \leq 4 \cdot 10^6, \quad i = 1, \ldots, 4$$

leads to

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	$3.55248 \cdot 10^{-21}$	1	1	5	5946	183	21.400
3	2	0	$3.55248 \cdot 10^{-21}$	1	1	5	5947	275	20.267
4	2	0	$3.55248 \cdot 10^{-21}$	1	1	5	5949	365	20.400
2	3	0	$2.28831 \cdot 10^{-20}$	1	1	5	3220	275	10.200
3	3	0	$2.28831 \cdot 10^{-20}$	1	1	5	3222	411	11.067
2	4	0	$7.62554 \cdot 10^{-18}$	1	1	4	5605	375	18.533

EXAMPLE 20: Matyas [36]

$$f_{Mat}(x) = 0.26\left(x_1^2 + x_2^2\right) - 0.48 x_1 x_2$$

$$-10 \leq x_i \leq 10, \quad i = 1, 2$$

$$f^* = 0, \quad x^* = (0, 0)$$

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	−0.06085	$9.76192 \cdot 10^{-52}$	1	1	7	91	87	0.600
3	2	−0.0025	$9.76192 \cdot 10^{-52}$	1	1	7	94	147	0.667
4	2	−0.000126953	$9.76192 \cdot 10^{-52}$	1	1	7	96	203	0.733
2	3	−0.0025	$2.13253 \cdot 10^{-55}$	1	1	6	113	147	0.667
3	3	$-6.33179 \cdot 10^{-5}$	$2.13253 \cdot 10^{-55}$	1	1	7	115	223	0.800
2	4	−0.000126953	$6.89361 \cdot 10^{-59}$	1	1	6	80	203	0.733

EXAMPLE 21: Schwefel No. 3.2 [36]

$$f_{S3.2}(x) = \sum_{i=2}^{3} \left[\left(x_1 - x_i^2 \right)^2 + (1 - x_i)^2 \right]$$

$$-10 \leq x_i \leq 10, \quad i = 1, \ldots, 3$$

$$f^* = 0, \quad x^* = (1, 1, 1)$$

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	$1.20614 \cdot 10^{-21}$	1	1	3	80	59	0.467
3	2	0	$1.20614 \cdot 10^{-21}$	1	1	4	82	111	0.533
4	2	0	$1.20614 \cdot 10^{-21}$	1	1	4	84	163	0.667
2	3	0	$1.68341 \cdot 10^{-19}$	1	1	3	95	111	0.600
3	3	0	$1.68341 \cdot 10^{-19}$	1	1	4	96	189	0.667
2	4	0	$4.29246 \cdot 10^{-18}$	1	1	4	89	163	0.667

EXAMPLE 22: Rosenbrock [36]

$$f_{RB}(x) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2$$

$$-5 \leq x_i \leq 5, \quad i = 1, 2$$

$$f^* = 0, \quad x^* = (1, 1)$$

n_{it}	n_d	$\underline{F}^*$	$\overline{F}^*$	n_M	n_L	l_s	n_{rf}	n_{if}	t
2	2	0	$1.36492 \cdot 10^{-22}$	1	1	4	101	31	0.267
3	2	0	$1.36492 \cdot 10^{-22}$	1	1	4	102	55	0.333
4	2	0	$1.36492 \cdot 10^{-22}$	1	1	4	104	77	0.400
2	3	0	$1.14764 \cdot 10^{-22}$	1	1	4	122	57	0.400
3	3	0	$1.14764 \cdot 10^{-22}$	1	1	4	124	89	0.467
2	4	0	$2.29523 \cdot 10^{-21}$	1	1	3	119	83	0.400

5. Conclusion

In this paper we have presented a branch and bound algorithm for a global optimization problem with bound constraints. One of the most important aspects of this algorithm is the strategy which is used for incorporating local optimization algorithms. This is done by using inclusion functions for improving starting points, and by incorporating a special scheme for calling the local optimization algorithm. Numerical results for many well-known problems demonstrate that at the very beginning approximations of a global minimum point and the global minimum value are calculated, and that for most test problems only between 1 and 3 local searches are performed. The bounds for the global minimum value and the global minimum points are proved to be correct; all sources of errors are taken into consideration. Moreover, our method requires no derivatives.

Examples 6–9 in Section 4 are the test problems proposed by Dixon und Szegö [6], [7] for the purpose of comparison of global optimization algorithms. In Törn and Žilinskas [35], the following times (standard unit time) for other global optimization algorithms are given.

Algorithm	Problem						
	BR	GP	S5	S7	S10	H3	H6
Parv87(P)	2.2	4.1	4.6	3.1	3.6	1	1.9
Brem70	0.5	0.7	1.5*	1.5*	2	2*	3
Fagiuo78	5	0.7	7	9	13	5	100
Price78	4	3	14	20	20	8	46
Törn78	4	4	10	13	15	8	16
Boend80	1	1.3	3	5	8	2.5	5
Boend82	1	1.5	3.5	4.5	7	1.7	4.3
Timm84	0.25	0.15	1	1**	2	0.5	2
Roton87	1.6	2.1	3.4**	3.5**	4.0**	1.8	2.8
Zilin80a	27.5	25.5	122	160	170	99	161
DeBia78a	14	15	23	20	30	16	21
Snym87	—	0.2	1.1	1.3	2.0	0.6	1.3
Parv87(S)	17	5.4	7.1**	9.8**	12**	4	14**

*Only a local minimum was found

**Global minimum not always found (several experiments)

The times for our method are:

Algorithm	Problem						
	BR	GP	S5	S7	S10	H3	H6
Our Method	1.3	0.3	0.6	0.7	0.8	4.7	6.3

where the following guaranteed bounds are calculated:

Our Method	BR	GP	S5	S7	S10	H3	H6
$\overline{F}^*$	0.397887	3	−10.1532	−10.4049	−10.5364	−3.86278	−3.32237
$\underline{F}^*$	0.397887	3	−10.1559	−10.4144	−10.5501	−4.32854	−4.14692

This comparison shows that our method works very well although guaranteed bounds are calculated additionally.

We are currently investigating a modification of our algorithm for problems where derivatives are available. First results show that in many cases this modification leads to an acceleration, and very sharp bounds for the global minimum

value **and** the global minimum points are calculated. In our future work, we intend to generalize this algorithm to constrained global optimization problems.

Appendix. Symbol Index

A	list of calculated approximations
$\alpha, \beta, \gamma, \delta$	parameters of the procedure SEARCH
$d(X, Y)$	distance of two boxes $X, Y \in I(\mathbb{R}^n)$
f	objective function
F	inclusion function of f
$\underline{F}^*$	guaranteed lower bound for the global minimum value
$\overline{F}^*$	guaranteed upper bound for the global minimum value
$H(\tilde{x})$	expansion box around a local or global minimizer
$k_{\max}$	max. number of bisections
L	list of subboxes (only used internally)
l_s	maximal length of the lists S,L,A
$m(X)$	midpoint of X
n_d	max. number of bisections for each direction
n_{it}	total number of iterations
n_L	number of calls of the descent method
n_M	number of local or global minimizers found by our algorithm
n_{rf}, n_{if}	total number of real and inclusion function calls, respectively
p	permutation
S	list of subboxes with $X^* \subseteq \bigcup \left\{ Y \mid (Y, \underline{F}(Y)) \in S \right\}$
t	machine independent standard unit time
$w(X)$	width of X
$w_{rel}(X)$	relative width of X
$X^k \to x$	sequence (X^k) converges to x
$\square(X)$	interval hull of X

References

1. Alefeld, G. and Herzberger, J. (1983), *Introduction to Interval Computations*, Academic Press, New York.
2. Boender, C., Kan, A. R., Timmer, G. and Stougie, L. (1982), A stochastic method for global optimization. *Mathematical Programming*, **22**: 125–140.
3. Brent, R. P. (1973), *Algorithms for Minimization without Derivatives*, Prentice-Hall Inc., Englewood Cliffs, New Jersey.
4. Charalambous, C. and Bandler, J. W. (1976), Non-linear minimax optimization as a sequence of least pth optimization with finite values of p. *J. Comput. Syst. Sci.* **7**(4): 377–391.

5. Csendes, T. (1991), Test Results of Interval Methods for Global Optimization, 417–424, in E. Kaucher, S. M. Markov, G. Mayer, *Computer Arithmetic, Scientific Computation and Mathematical Modelling*, IMACS.
6. Dixon, L. C. W. and Szegö, G. P. (eds.), (1975), *Towards Global Optimization*, North-Holland, Amsterdam.
7. Dixon, L. C. W. and Szegö, G. P. (eds.), (1978), *Towards Global Optimization 2*, North-Holland, Amsterdam.
8. Hansen, E. R. (1979), Global Optimization Using Interval Analysis – the One-Dimensional Case, *J. Optim. Theor. and Appl.* **29**, 331–344.
9. Hansen, E. R. (1980), Global Optimization Using Interval Analysis – the Multidimensional Case, *Numerische Mathematik* **34**, 247–270.
10. Hansen, E. R. (1992), *Global Optimization Using Interval Analysis*, Marcel Dekker Inc., New York.
11. Horst, R. and Tuy, H. (1990), *Global Optimization*, Springer-Verlag, Berlin.
12. Jansson, C. (1991), A Global Minimization Method: The One-Dimensional Case, Bericht 91.2 des Forschungsschwerpunktes Informations- und Kommunikationstechnik der TU Hamburg-Harburg.
13. Jansson, C. (1992), A Global Optimization Method Using Interval Arithmetik. In L. Atanassova and J. Herzberger, *Computer Arithmetic and Enclosure Methods*, 259–267, North-Holland, Amsterdam.
14. Jansson, C. and Knüppel, O. (1992), A Global Minimization Method: The Multi-Dimensional Case, Bericht 92.1 des Forschungsschwerpunktes Informations- und Kommunikationstechnik der TU Hamburg-Harburg.
15. Kearfott, B. Du, K. (1993) The Cluster Problem in Global Optimization, *Computing Suppl.* **9**, 117–127.
16. Knüppel, O. (1993), BIAS — Basic Interval Arithmetic Subroutines, Bericht 93.3 des Forschungsschwerpunktes Informations- und Kommunikationstechnik der TU Hamburg-Harburg.
17. Knüppel, O. (1993), PROFIL — Programmer's Runtime Optimized Fast Interval Library, Bericht 93.4 des Forschungsschwerpunktes Informations- und Kommunikationstechnik der TU Hamburg-Harburg.
18. Knüppel, O. (1994), PROFIL/BIAS — A Fast Interval Library, *Computing* **53**, 277–287.
19. Kulisch, U. and Miranker, W. L. (1981), *Computer Arithmetic in Theory and Practice*, Academic Press, New York.
20. Lohner, R. (1989), Enclosing all eigenvalues of symmetric matrices. In *Accurate Numerical Algorithms*, A Collection of Research Papers, volume 1 of Research Reports ESPRIT, Project 1072, DIAMOND, 87–103. Springer, Berlin.
21. Moore, R. E. (1966), *Interval Analysis*, Prentice-Hall, Englewood Cliffs, N.J.
22. Moore, R. E. (1976) On Computing the Range of Values of a Rational Function of n Variables over a Bounded Region, *Computing* **16**, 1–15.
23. Moore, R. E. (1979), *Methods and Applications of Interval Analysis*, SIAM, Philadelphia.
24. Moore, R., Hansen, E., and Leclerc, A. (1992), Rigorous Methods for Global Optimization. In *Recent Advances in Global Optimization*, Princeton series in computer science, 321–342. Princeton University Press, Princeton, New Jersey.
25. Murty, K. G. and Kabadi, S. N. (1987), Some NP-Complete Problems in Quadratic and Nonlinear Programming, *Mathematical Programming* **39**, 117–130.
26. Neumaier, A. (1990), *Interval Methods for Systems of Equations*, Cambridge University Press.
27. Pardalos, P. M. and Rosen, J. B. (1987), *Constrained Global Optimization: Algorithms and Applications*, Springer Lecture Notes Comp. Sci. 268, Berlin.
28. Ratschek, H. (1985), Inclusion Functions and Global Optimization, *Mathematical Programming* **33**, 300–317.
29. Ratschek, H. and Rokne, J. (1984), *Computer Methods for the Range of Functions*, Ellis Horwood Limited, Chichester.
30. Ratschek, H. and Rokne, J. (1988), *New Computer Methods for Global Optimization*, Ellis Horwood Limited, Chichester.

31. Ratz, D. (1992), An Inclusion Algorithm for Global Optimization in a Portable PASCAL-XSC Implementation, in L. Atanassova and J. Herzberger, *Computer Arithmetic and Enclosure Methods*, North-Holland, 329–339, Amsterdam.
32. Rump, S. M. (1983), Solving Algebraic Problems with High Accuracy, in U. W. Kulisch and W.L. Miranker (eds), *A New Approach to Scientific Computation*, Academic Press, New York.
33. Shen, Zuhe, Neumaier, A., and Eiermann, M.C. (1990), Solving Minimax Problems by Interval Methods, *BIT* **30**, 742–751.
34. Skelboe, S. (1974), Computation of Rational Interval Functions, *BIT* **14**, 87–95.
35. Törn, A. and Zilinskas, A. (1989), *Global Optimization*, Springer-Verlag, Berlin Heidelberg New York.
36. Walster, G., Hansen, E., and Sengupta, S., (1985), Test results for a global optimization algorithm. *Numerical Optimization* 1984, 272–287.
37. Wilkinson, J. H. (1971), Modern error analysis, *SIAM Rev.* **13**, 548–568.