A New Method for Global Optimization Using Hyperbolic Cross Points

Erich Novak and Klaus Ritter
Mathematisches Institut
Universität Erlangen-Nürnberg
91054 Erlangen
Germany

Abstract. We propose a new numerical method for finding the global minimum of a function $f : [0,1]^d \to \mathbf{R}$. Our method is based only on function values at the hyperbolic cross points and uses an adaptive order of these points. We motivate our method by complexity results and also give numerical examples.

1. Introduction

We discuss the following problem. Let

$$[a,b] = \prod_{i=1}^{d} [a_i, b_i]$$

be a given box (or hyperrectangle) in $\mathbf{R}^d$. For a function $f : [a,b] \to \mathbf{R}$ we wish to find a point $x_* \in [a,b]$ such that

$$f(x_*) \approx \inf_{x \in [a,b]} f(x).$$

The only assumption on f is that

$$\{x \in [a,b] \mid f(x) < \inf_{x \in [a,b]} f(x) + \varepsilon\}$$

contains an open set for every $\varepsilon > 0$. In particular, this is true if f is continuous or upper semicontinuous.

We allow numerical methods which use function values at sequentially chosen knots $x^1, \ldots, x^n$. The first knot x^1 is independent of f and fixed for a particular method. The knots x^k with $k > 1$, however, may depend on the previously computed data. They are computed according to functions $\Psi_k : \mathbf{R}^{k-1} \to [a,b]$ via

$$x^k = \Psi_k(f(x^1), f(x^2), \ldots, f(x^{k-1})).$$

We use a given oracle or subroutine to evaluate $f(x^k)$ once the knot x^k is chosen. After a finite number of steps, say n, the information

$$N_n(f) = (f(x^1), f(x^2), \ldots, f(x^n))$$

about f is used to construct the point

$$x_* = \Psi_*(N_n(f))$$

according to a function $\Psi_* : \mathbf{R}^n \to [a,b]$. The error of the method, applied to f, is then given by

$$\Delta_n(f) = f(x_*) - \inf_{x \in [a,b]} f(x).$$

In our method and in many other methods the information $N_n(f)$ is used in a very simple way: we take any $x_* = x_*^n \in \{x^1, \ldots, x^n\}$ such that

$$f(x_*^n) = \min\{f(x^1), \ldots, f(x^n)\}$$

as the approximation to the true minimum. The main problem therefore is a good choice of the functions Ψ_k that define the knots used by a specific method.

In this paper we suggest to use hyperbolic cross points, starting with the midpoint of the box as x^1. Thereafter hyperbolic cross points are chosen adaptively, with preference to regions where small function values have been found or where a comparatively small number of samples has been taken.

Remarks.

1) It is easy to define methods with

$$\lim_{n \to \infty} \Delta_n(f) = 0$$

for all continuous f. It is sufficient (and also necessary) to guarantee that the sequence $(x^n)_n$ is dense in $[a,b]$ for any f. We want that $\Delta_n(f)$ is "small for nice functions". Therefore the sheer convergence of the method is not sufficient for its efficiency.

2) The only information on f which is used by our method is given by a subroutine that allows to compute the function values $f(x^k)$. We do not use values of derivatives of f. We also do not use a bound of the Lipschitz constant (or of any other norm) of f. Of course we agree that such an additional information on f may be very useful. However, such an information often is not available and hence we design an algorithm that works reasonably without such an information. Results and further references on Lipschitz optimization may be found in Hansen and Jaumard (1995).

3) There are still other algorithms using interval arithmetic. These methods use more and different information about f and are a mixture between symbolic and numerical methods. Interval methods often yield very good results if they can be applied. See Hansen (1992) and Ratschek and Rokne (1995).

4) It is possible, in particular, that the functions Ψ_k are constants, i.e., the kth knot x^k does not depend on function values already computed and we use the same sequence $(x^k)_k$ for all functions. Methods of this simple form are usually called nonadaptive or passive, in contrast to adaptive or active methods which are defined by non-constant Ψ_k's. We will propose an adaptive method in Section 3, but consider adaptive as well as nonadaptive methods in Section 2.

2. Some known complexity results

The problem of global optimization is a difficult one. No numerical method exists that is good for all reasonable functions f. These statements can be made precise by lower bounds – valid for all methods.

We explain the essence of such complexity results. According to the previous section, any method for global optimization is specified by $\Psi_1 = x^1$ together with the mappings Ψ_k and Ψ_*. We use the notation

$$\Psi = (\Psi_1, \ldots, \Psi_n, \Psi_*).$$

In this section we apply methods Ψ only to functions from particular classes F, and Ψ may be tuned to give good results on this class. This means that (only in this section) we know a priori that $f \in F$. This usually includes the knowledge of Lipschitz constants or other norms of f.

The number

$$\Delta_n(\Psi, F) = \sup_{f \in F} \Delta_n(f)$$

is called the maximal error of Ψ on the class F. Minimizing over all methods which use n function values we get

$$e_n(F) = \inf_{\Psi} \Delta_n(\Psi, F),$$

and the sequence $e_n(F)$ indicates the (worst case) complexity of finding the global minimum for the class F.

The following result is well known, see Nemirovsky and Yudin (1983), Novak (1988), Wasilkowski (1984, 1992) and Vavasis (1991) for this and for more general results. Recall that d denotes the dimension of the box $[a, b]$. By $D^r f$ with $r = (r_1, \ldots, r_d)$ we denote a partial derivative and we put $|r| = \sum r_i$, as usual.

Theorem 1. Let F be a Hölder class of the form

$$F = \{f : [a, b] \to \mathbf{R} \mid |D^r f(x) - D^r f(y)| \le \|x - y\|^\alpha \text{ for all } r \text{ with } |r| \le k\},$$

where $0 < \alpha \le 1$ and $k \in \mathbf{N}_0$. Then there is a constant $c > 0$ such that

$$e_n(F) > c \cdot n^{-(k+\alpha)/d}.$$

This lower bound is sharp and can be achieved with a passive algorithm.

Remarks.
This is a "negative" result. Any clever adaptive algorithm cannot be better than the simplest nonadaptive method, based on a regular grid. We stress that the lower bound is still valid if we allow to use derivatives of order at most k.

Lipschitz optimization, in particular, is an intractable problem for $d >> 1$. Here we have $k = 0$ and $\alpha = 1$ and obtain

$$e_n(F) > c \cdot n^{-1/d}.$$

In this and some other cases it is possible to specify the optimal constant c.

It is not possible to obtain better results by using stochastic or Monte Carlo methods. This may be surprising because stochastic search methods are often used in practice. It is known, however, that the complexity result of Theorem 1 is also true if we allow stochastic methods. See Novak (1988) and Wasilkowski (1989) for details. So it seems that stochastic methods do not provide better solutions for the difficulties of global optimization and we do not discuss them further.

What are possible consequences from Theorem 1? We think there are two.

Consequence 1: We have to look for "smaller" classes F of functions of several variables that allow smaller errors and still contain many interesting functions.

Consequence 2: Instead of a worst case analysis, we can opt for an average case analysis. Is it possible to find a method that is good for "most" $f \in F$?

We give some comments concerning both consequences.

Ad 1: We often deal with "nonisotropic" situations where some directions in $\mathbf{R}^d$ are more important than others. Many interesting functions can be written as the sum of functions, depending on only a few of the variables. We give an example. Assume that f is of the form

$$f(x) = \sum_{i,j=1}^{d} f_{i,j}(x_i, x_j), \qquad (1)$$

i.e., f is the sum of functions $f_{i,j}$ depending only on two variables. Let us assume that $f_{i,j} \in C^2([a,b])$ for all pairs (i,j). By classical results, such as Theorem 1, we can use methods Ψ based on a regular grid that guarantee an error

$$\Delta_n(\Psi, C^2([a,b])) \leq c \cdot n^{-2/d}.$$

We obtain a very slow convergence if d is large. A function of the form (1) is $C^2([a,b])$ but, in addition, all derivatives of the form

$$D^r f, \qquad \text{with } r_i \leq 1 \text{ for every } i$$

exist. We define $W_\infty^{\mathbf{1}}([a,b])$ by

$$W_\infty^{\mathbf{1}}([a,b]) = \{ f : [a,b] \to \mathbf{R} \mid \| D^r f \|_\infty \leq 1 \text{ for all } r \text{ with } r_i \leq 1 \ \}.$$

Using results of Temlyakov (1987), see also Woźniakowski (1992), one can show that there are methods for $W_\infty^{\mathbf{1}}([a,b])$ with error bounds that only weakly depend on the dimension d. One gets an error bound

$$\Delta_n(\Psi, W_\infty^{\mathbf{1}}([a,b])) \leq c \cdot n^{-1} (\log n)^{2(d-1)}.$$

It is quite remarkable, however, that this bound cannot be achieved using grid points – grid points only yield the poor bound $n^{-1/d}$. Instead one can use a method Ψ which is based on hyperbolic cross points, see Section 3.

Ad 2: Adaptive methods are not better than nonadaptive methods in a worst case sense on F. This is true for any class F that is symmetric, i.e., $f \in F$ implies $-f \in F$, and convex, see Novak (1988) and Wasilkowski (1984) for details. In practice, however, we usually use adaptive methods and believe that they are much better on the average. The superiority of adaptive methods on the average is difficult to prove but supported by many numerical examples.

To compute the average error one can take the classical Wiener measure on $C([0,1])$ for $d = 1$ and the Wiener sheet measure if $d > 1$. Ritter (1990) obtained tight bounds for the average error of optimal nonadaptive methods in the case $d = 1$, other results are contained in Wasilkowski (1992), Al-Mharmah and Calvin (1994), and Calvin (1994). It is clear from Ritter's results, and also follows by "numerical evidence", see Nestel (1991), that adaptive methods are better on the average than nonadaptive ones. It is not known, however, what are optimal methods with respect to the average error. Intuitively, a good adaptive method should place "many" points where f probably is small. There exist several interesting algorithms that are based on the Bayesian approach, see Mockus (1989), Törn and Žilinskas (1989), and Boender and Romeijn (1995). Average error bounds for these methods, however, seem to be unknown.

3. The new algorithm: competing hyperbolic cross points

Our basic idea is the following: Use hyperbolic cross points which allow good worst case error bounds for the class $W^{\mathbf{1}}_{\infty}([a,b])$. However, we want to improve the non-adaptive algorithm (based on these points) to get an algorithm that should be much better on the average. Therefore we do not use the hyperbolic cross points in their natural order but use an adaptive order – based on the previous function values.

We now explain our algorithm in more detail. It is convenient to use the box

$$[a,b] = \prod_{i=1}^{d} [-0.5,\, 0.5]$$

as the domain of f. A general hyperrectangle will be transformed by an affine mapping onto this special box. Other transformations apply to more complicated domains. We first explain the notion of a hyperbolic cross point. Any point $x = (x_1, \dots, x_d) \in [-0.5,\, 0.5]$ with dyadic coordinates

$$x_i = \pm \sum_{j=1}^{k} a_j \, 2^{-j}, \qquad a_j \in \{0,1\} \tag{2}$$

is a hyperbolic cross point. The representation (2) of x_i is unique if we assume that $a_k = 1$. In that case the number k is called the level of x_i. We also say that the level of the point 0 is 0. Finally the level of a hyperbolic cross point x is defined by

$$\mathrm{level}\,(x) = \sum_{i=1}^{d} \mathrm{level}\,(x_i).$$

Hence we have, for example, $\mathrm{level}\,(0.25,\, 0.5) = 3$ and $\mathrm{level}\,(0,\, 0.375) = 3$. The basic nonadaptive algorithm is the following. Compute $f(x)$ for all hyperbolic cross points x with $\mathrm{level}\,(x) \le k$ and take x^n_* such that

$$f(x^n_*) = \min(f(x^1), \dots, f(x^n)).$$

We now describe our adaptive refinement of this nonadaptive algorithm. We need the notion of the neighbors of degree l of a hyperbolic cross point x. Assume that $\mathrm{level}\,(x) = k$. Then every neighbor of x (of any degree) only differs in one coordinate from x. The neighbors of degree m all have the level $k+m$ and are the nearest points – in each of the $2d$ directions of the x_i-axes – with that property. To give an example, the neighbors of degree 1 of $(0.5,\, 0)$ are $(0.5,\, \pm 0.5)$ and $(0.25,\, 0)$. The neighbors of degree 2 of $(0.5,\, 0)$ are $(0.5,\, \pm 0.25)$ and $(0.375,\, 0)$.

Now our algorithm works as follows. First put $x^1 = 0$ because $x = 0$ is the only point with the lowest level 0. One further step of the algorithm consists in choosing all (up to $2d$) neighbors of a certain degree of one point $\tilde{x}$. Assume that we already have computed the values of f at the knots $x^1, x^2, \ldots, x^k$. We have to explain how to choose

$$\tilde{x} \in \{x^1, \ldots, x^k\} = X$$

and which neigbhors of $\tilde{x}$ are taken as the next knots. We take $\tilde{x} \in X$ as the best point in X with respect to some goodness-criterion g. To define g, we need three numbers.

First, the level of the points $x^i \in X$ is important, because we want to prefer points with a small level. Second, we count how often each point x^i already has been the best point, chosen in earlier steps of the algorithm. This number plus 1 is called the degree of x^i. The third number is the rank of x^i,

$$\mathrm{rank}\,(x_i) = \#\{y \in Y \mid y \le f(x_i)\}$$

where

$$Y = \{f(x^1), f(x^2), \ldots, f(x^k)\}.$$

Hence $\mathrm{rank}\,(x_i) = 1$ iff $f(x_i)$ is minimal in Y and $\mathrm{rank}\,(x_i) = \#Y \le k$ if $f(x_i)$ is maximal in Y.

We define the quality $g(x^i)$ by

$$g(x^i) = (\mathrm{level}\,(x^i) + \mathrm{degree}\,(x^i))^\alpha \cdot \mathrm{rank}\,(x^i)^{1-\alpha}$$

and stress that only the level is fixed while the degree and the rank of a point are changing during the computation. Here

$$\alpha \in [0, 1]$$

is a parameter of our algorithm. The value $\alpha = 1$ means that the method is purely nonadaptive, while $\alpha = 0$ means that the algorithm highly prefers a local search.

In each step we determine $\tilde{x} \in X$ as the point where g takes the smallest value. The next knots are the neighbors of $\tilde{x}$ of the present degree of $\tilde{x}$. After these neighbors are chosen as the next knots we increase the degree of $\tilde{x}$ by one and compute the new ranks of each knot. In order to do so, we first must compute the function values at the new knots.

This is – basically – our algorithm. There are two further modifications. First, we only consider points x with coordinates

$$x_i = \pm \sum_{j=1}^{k} a_j \, 2^{-j}, \qquad a_j \in \{0, 1\},$$

where

$$k \le k_0.$$

The number k_0 is another parameter of our algorithm, called fineness. For each coordinate we have $2^{k_0} + 1$ points, the total grid consists in

$$N = (2^{k_0} + 1)^d$$

points. We are only interested in a global search and therefore use a rather small number N. As a consequence, our method usually will not find the global optimum exactly.

There is one last modification. If the algorithm finds a point x^i such that x^i is a "local minimum", i.e., $f(x^i)$ is smaller than all nearest neighbors (with distance 2^{-k_0}), than we do not allow that in any further step a best point $\tilde{x}$ is too close to x^i. To this end we exclude a box of sidelength

$$0 \leq \varepsilon < 1$$

with center x^i when minimizing the quality g in all further steps. The number ε is another parameter, see the following comment.

A comment on parameters. As described above, our algorithm uses the parameters α, ε, and the fineness k_0 of the grid. Of course, there is one more parameter, the number n of function values. The number n must be chosen larger if f is a "difficult" function or if d is large.

Often our method works good only with a good choice of the parameters. Experienced users can also use their a priori knowledge about a specific function to adjust the parameters. We feel, however, that this is not adequate in many cases: if one applies an algorithm then one usually wants to use a small number n of function evaluations – and it is not meaningful that there is an unknown choice of certain parameters that will work fine.

Therefore we use certain standard parameters for α, ε, and k_0, depending on the dimension. We use, for example, a smaller fineness k_0 in higher dimension. This means that we can get more exact results in low dimension. One can define very difficult functions in higher dimensions but we choose our standard parameters in a way that simpler problems can be solved with a low amount of function values – yielding a low accuracy. So, if we use the standard parameters, there is only one parameter, the number n of function evaluations. Of course n should depend on the "difficulty" of f and/or on the available resources. The definition of the standard parameters is as follows:

d	k_0	ε	α
2	14	0.02	0.5
3	7	0.05	0.4
4	5	0.08	0.2
5	4	0.15	0.15
≥ 6	3	0.3	0.1

The choice of this parameters can be justified by theoretical considerations, at least in part. It may suffice, however, to note that this choice of parameters is good for most examples we have tried so far. This is true in particular for the seven "standard test functions", see Section 4, and for other "simple" examples. For difficult problems we must use a finer grid (larger value of k_0), a smaller value of ε and a larger value of α. But then also the number n of function evaluations usually must be increased, sometimes dramatically.

4. Some numerical results

Since the work of Dixon and Szegö (1978) there exist seven test examples used very often in the literature. See also, for example, Törn and Žilinskas (1989), Rinnoy Kan and Timmer (1985), Breiman and Cutler (1993), and Byrd et al. (1990). The informed reader will know that not all standard methods work for all these test examples. Also some methods require a special choice of parameters with different values for each example.

Results for seven standard test functions. In the following we present our results, we always used the standard parameters, as explained in Section 3. We do not define the functions and use the notation of Törn and Žilinskas (1989).

function	n	$f(x_*)$	x_*
BR	220	0.397	(9.422, 2.473)
GP	9	3.000	(0.000, 1.000)
H3	85	-3.862	(0.125, 0.555, 0.852)
S5	121	-8.806	(4.063, 4.063, 4.063, 4.063)
S7	478	-9.055	(4.063, 4.063, 4.063, 4.063)
S10	439	-9.190	(4.063, 4.063, 4.063, 4.063)
H6	580	-3.035	(0.125, 0.125, 0.500, 0.250, 0.375, 0.625)

Of course, for the function **GP** it is just luck that we find the exact global minimum after 9 evaluations. For similar functions of the same structure we would typically need 100 to 200 values. For the Shekel functions **S5**, **S7**, and **S10**, the true minima are approximately -10.153, -10.402, and -10.536. Hence above results are not very accurate. This is because our grid is not very fine, we have $d = 4$ and $k_0 = 5$. Nevertheless we have found a point x_* near the true minimum, because all other local minima have values greater than -5.2. One could easily get better results by applying a local refinement after the global search. Our C-program contains such an option. For the function **H6**, where $d = 6$ and $k_0 = 3$, our grid is even coarser. The global minimum is approximately -3.32. Again, x_* lies within the region of attraction of the global minimum, and local methods could be used to get a better approximation.

Remarks. Compared to other algorithms our results are not too bad. We want to stress, however, that some methods are better for some of the examples, at least for two reasons:

a) The test functions are not typical elements of the class $W^{\mathbf{1}}_{\infty}([a,b])$, most of them would more naturally be considered in isotropic classes. The Shekel functions, for example, are of the type

$$f(x) = -\sum_{i=1}^{m} \frac{1}{\|x-d_i\|^2 + c_i}$$

(with $m = 5, 7, 10$ and $d = 4$).

b) In all seven test examples, the relative size of the region of attraction of the global minimum is very large, between 0.2 and 1, see Törn and Žilinskas (1989). This is why standard multistart-algorithms have a good chance to find the global minimum with only a few starting points.

Because of these special properties of the seven test examples we also used other examples.

An example of Mladineo (1992). This example is defined by

$$f_d(x) = 1 + \frac{1}{d}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} \cos(10\log((i+1)x_i)),$$

where $[a,b] = [0.01,\, 1]^d$. Here the region of attraction of the global minimum is very small. First we discuss the case $d = 2$. With the standard parameters of our algorithm we need $n = 16418$ values to find the point $x_* = (0.0158,\, 0.0105)$ with $f(x_*) = 0.000484$. The true minimum is near $(0.011527,\, 0.014405)$ and its value is approximately 0.000170. Although x_* is rather close to the true minimum, it is not within the region of attraction of the global minimum.

We believe that this is a difficult problem for most methods. We found this problem in Hansen and Jaumard (1995), where Lipschitz optimization is discussed. Assume that we know the Lipschitz constant $L = 988.82$ of this function. Then one can guarantee that the error is bounded by some given $\delta > 0$ using different known methods. For the specific error bound $\delta = 0.0692$ all the methods discussed in Hansen and Jaumard (1995) need more than 10^6 function evaluations. In contrast, we use $n = 16418$ points to obtain an error of about 0.0003. Furthermore our algorithm does not assume the knowledge of a Lipschitz constant. It should be stressed, however, that we do not get any error bound – and it is clear that no error bound is possible without assuming some a priori information about f, such as a Lipschitz constant.

Now we discuss the case $d = 3$. This example is very difficult for the methods in Lipschitz-optimization described in Hansen and Jaumard (1995). These authors used more than 10^6 points and did not reach the region of attraction of the global minimum.

Instead they found a point $x_* = (0.040555, 0.069074, 0.052778)$ with $f(x_*) \approx 0.015$. To guarantee an error $\delta = 0.8$, these methods would probably need more than 10^9 function evaluations. Observe that $\delta = 0.8$ is a very modest accuracy because it is clear that $f(x) \in [0,3]$ for all $x \in [a,b]$.

If we use our algorithm with the standard parameters and $n = 13667$, then we obtain $x_* = (0.040937, 0.180156, 0.133750)$ with $f(x_*) = 0.0318$. The error is about 0.0316 and the point x_* is far from the region of attraction of the global minimum. The standard parameters are not sufficient to solve this difficult problem – the fineness $k_0 = 7$ defines a grid that is much too coarse for this example. If we have enough resources then we can of course use a finer grid and a larger number of knots.

Simplified Griewank functions. Griewank (1981) considers two examples for dimension $d = 2$ and $d = 10$, respectively. These examples are similar as the last example of Mladineo, with an extremely high number of local minima in the case $d = 10$. We study a family f_d of simplified functions defined by

$$f_d(x) = 1 + \frac{1}{d}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} \cos(x_i).$$

As the domain of f_d we take the box $[a,b] = [-1,4]^d$. The optimal solution of course is $x = 0$ with $f(x) = 0$.

Again we used our standard parameters, i.e., $k_0 = 3$ for $d \geq 6$. Then, because of our special choice of the box, the point $x = 0$ is not in our grid and the best possible x_* has coordinates $x_* = (0.25, \ldots, 0.25)$. It is easy to find this point by our algorithm. For dimension $d = 30$, for example, we just need 2700 function evaluations to find this point. For $d = 10$ we need 300 points and for $d = 20$ we need 1200 points. Probably one can prove that our algorithm needs exactly $n = 3d^2$ points (if $d \geq 6$, i.e., if we take $k_0 = 3$). Although the point x_* is not very accurate it can be used as a starting point for any local method to find the global minimum $x = 0$. One could also argue that the whole box has volume 5^d, while the volume of those points x with $|x_i| \leq \frac{1}{4}$ for all i is only $(\frac{1}{2})^d$.

5. Concluding remarks

We believe that our method is promissing. We did some programming in C, the programs are available from the authors, preferably by electronic mail (novak@mi.uni-erlangen.de or ritter@mi.uni-erlangen.de). It goes without saying that the authors would like to be informed about results and modifications concerning this algorithm.

A word of warning. So far we did not make the programming very carefully. Using more advanced data structures it should be possible to get a much faster routine. This is not crucial, however, for small problems, where we need at most a few thousands of function values.

References

H. Al-Mharmar and J. Calvin (1994): Optimal random non-adaptive algorithm for global optimization of Brownian motion. Preprint, Georgia Institute of Technology, Atlanta.

C. G. E. Boender and H. E. Romeijn (1995): Stochastic methods. In: Handbook of Global Optimization, R. Horst, P. M. Pardalos, eds., Kluwer, Dordrecht, 829–869.

L. Breiman and A. Cutler (1993): A deterministic algorithm for global optimization. Math. Programming **58**, 179–199.

R. H. Byrd, C. L. Dert, A. H. G. Rinnoy Kan, and R. B. Schnabel (1990): Concurrent stochastic methods for global optimization. Math. Programming **46**, 1–29.

J. Calvin (1994): Average performance of passive algorithms for global optimization. To appear in J. Math. Anal. Appl.

L. C. W. Dixon and G. P. Szegö (1978): The global optimisation problem: an introduction. In: Towards Global Optimisation, Vol. 2, L. C. W. Dixon, G. P. Szegö, eds., North-Holland, Amsterdam, 1–15.

A. O. Griewank (1981): Generalized decent for global optimization. J. Optimization Th. Appl. **34**, 11–39.

E. Hansen (1992): Global Optimization Using Interval Analysis. Dekker, New York.

P. Hansen and B. Jaumard (1995): Lipschitz optimization. In: Handbook of Global Optimization, R. Horst, P. M. Pardalos, eds., Kluwer, Dordrecht, 407–493.

R. H. Mladineo (1992): Stochastic minimization of Lipschitz functions. In: Recent Advances in Global Optimization, C. A. Floudas, P. M. Pardalos, eds., Princeton University Press, Princeton, 369–383.

J. Mockus (1989): Bayesian Approach to Global Optimization. Kluwer, Dordrecht.

A. S. Nemirovsky and D. B. Yudin (1983): Problem Complexity and Method Efficiency on Optimization. Wiley, Chicester.

F. Nestel (1991): Globale Optimierung auf Wienerpfaden. Thesis, Erlangen.

E. Novak (1988): Deterministic and Stochastic Error Bounds in Numerical Analysis. Lecture Notes in Mathematics **1349**, Springer, Berlin.

H. Ratschek and J. Rokne (1995): Interval methods. In: Handbook of Global Optimization, R. Horst, P. M. Pardalos, eds., Kluwer, Dordrecht, 751–828.

A. H. G. Rinnooy Kan and G. T. Timmer (1985): A Stochastic Approach to Global Optimization. In: Numerical Optimization 1984, P. T. Boggs, R. H. Byrd, eds., SIAM, Philadelphia, 245–262.

K. Ritter (1990): Approximation and optimization on the Wiener space. J. Complexity **6**, 337–364.

V. N. Temlyakov (1987): Approximate recovery of periodic functions of several variables. Math. USSR Sbornik **56**, 249–261.

A. Törn and A. Žilinskas (1989): Global Optimization. Lecture Notes in Computer Science **350**, Springer, Berlin.

S. A. Vavasis (1991): Nonlinear Optimization, Complexity Issues. Oxford University Press, New York.

G. W. Wasilkowski (1984): Some nonlinear problems are as easy as the approximation problem, Comput. Math. Appl. **10**, 351–363.

G. W. Wasilkowski (1989): Randomization for continuous problems. J. Complexity **5**, 195–218.

G. W. Wasilkowski (1992): On average complexity of global optimization problems. Math. Programming **57**, 313–324.

H. Woźniakowski (1992): Average case complexity of linear multivariate problems, Part 1 and 2. J. Complexity **8**, 337–392.