Baltic J. Modern Computing, Vol. 3 (2015), No. 3, pp. 214–224

A Two-Phase Global Optimization Algorithm for Black-Box Functions

Gražina GIMBUTIENĖ, Antanas ŽILINSKAS

Institute of Mathematics and Informatics, Vilnius University, Akademijos str. 4, LT-2600 Vilnius, Lithuania

grazina.gimbutiene@mii.vu.lt, antanas.zilinskas@mii.vu.lt

Abstract. A modification of the global optimization algorithm, rooted in the statistical theory of global optimization, is proposed. The original method is based on the hyper-rectangular partition of the feasible region where a hyper-rectangle for subdivision is selected using a criterion related to the probability of improvement. The idea of the modification is in the coordination of local and global phases of search. The testing results show that the proposed modification improves the performance of the original algorithm.

Keywords: global optimization, statistical models, two-phase algorithm

1 Introduction

Global optimization problems are among the most difficult in optimization theory as well as in applications. The development of global optimization algorithms is very active in accordance with both, heuristic and mathematical, approaches. The origins of heuristic algorithms are really diverse: from the principle "survives the fittest" to the simulation of harmony search in music. Theoretical approaches are based on the two basic models: deterministic and statistical. In the former case, the assumption on bounded variation of the function values is crucial, (Floudas, 2000), (Horst et al., 1995), (Pardalos and Romeijn, 2002), (Pinter, 1996). The most popular model in this class is Lipschitzian, see (Sergeyev and Kvasov, 2006, 2011), (Horst et al., 1995), (Pinter, 1996). The deterministic models are inherently oriented to the guaranteed result, and the respective algorithms are developed in the view of the worst case behavior of an objective function. The algorithms based on the statistical approach are oriented to the average case where the unknown objective function values are interpreted as random variables (Mockus, 1989), (Strongin and Sergeyev, 2000), (Zhigljavsky and Žilinskas, 2008).

In the present paper an algorithm is developed which is rooted in the statistical models based theory of global optimization. In the original P-algorithm, for the current computation of the objective function value a point is selected where the improvement

is most probable. A version of the hybridization of the idea of the P-algorithm with the decomposition of the feasible region is substantiated in (Calvin et al., 2015). The idea of decomposition of the feasible region is widely used in global optimization, see e. g. (Calvin and Žilinskas, 2014), (Paulavičius et al., 2014), (Sergeyev and Kvasov, 2006), (A. Žilinskas and J. Žilinskas, 2002). However, differently from the most similar algorithms (Calvin and Žilinskas, 2014) and (A. Žilinskas and J. Žilinskas, 2002), the feasible region in (Calvin et. al., 2015) is divided not into simplices, but into hyper-rectangles. A hyper-rectangle for the subdivision is selected by the maximization of the criterion related to the improvement probability. The high asymptotic convergence rate of this algorithm is shown in (Calvin et al., 2015). However, from an application point of view it is important to have an acceptable precision after a not too large number of iterations. To achieve a better, in the mentioned sense, performance a heuristic modification is implemented: two phases of the search are defined. The desirable improvement is pursued by the alternation of the local and global phases of the search. The idea of this modification is borrowed from (Paulavičius et al., 2014). The testing results show that the desirable improvement was achieved.

2 The proposed algorithm

2.1 Statistically justified partitioning-based approach

The goal of the presented research was to create a global optimization algorithm assuming that the available information on the objective function is scarce, but the algorithm nevertheless possess the following properties:

- The search strategy based on the theory of rational decisions under uncertainty,
- The theoretically established asymptotic convergence rate,
- The implementation complexity similar to that of the deterministic algorithms of similar purpose,
- The competitiveness with other algorithms with respect to the precision after a modest number of iterations.

We consider the minimization problem $\min_{x\in\mathbf{A}} f(x)$, where little is known about $f(\cdot)$, and $\mathbf{A} \subset \mathbb{R}^d$ is a hyper-rectangle. The uncertainty about $f(x)$ is typical, for example, in the case where the values of $f(x)$ are computed by an unfamiliar software, and properties of $f(\cdot)$, such as non-convexity and multi-modality, cannot be excluded. To justify a search strategy in the described situation of uncertainty, a "rational optimizer" should define a model of uncertainty, for example, like a statistical model of uncertainty in the theory of expected utility (Fishburn, 1970). Let us consider the current minimization step, where N function values have been computed at previous steps: $y_i = f(x_i)$, $i = 1, \ldots, N$. A rational choice of a point for the next computation of the objective function value cannot be performed without an assessment of the uncertainty in the result of that computation. The very general assumptions on the rational perception of uncertainty imply a random variable model of the value of the objective function; that is, those assumptions imply a random variable ξ_x as a model of $f(x)$ for $x \neq x_i, i = 1, \ldots, N$, where parameters of the distribution of ξ_x depend on

$x_i,\ y_i = f(x_i),\ i = 1, \ldots, N$; see (Zhigljavsky and Žilinskas, 2008). Various strategies of the global search have been developed using statistical models; we refer to (Mockus, 1989), (Strongin and Sergeyev, 2000) and (Zhigljavsky and Žilinskas, 2008) for their description and analysis. The so-called P-algorithm selects for the next computation of the objective function value the point where the improvement is most probable:

$$x_{N+1} = \arg\max_{x \in \mathbf{A}} \mathbf{P}(\xi_x < y_{0N} - \epsilon), \tag{1}$$

where $y_{0N} = \min_{1 \le i \le N} y_i$, $m(x|x_i, y_i, i = 1, \ldots, N)$ is the expected value of ξ_x, and $s(x|x_i, y_i, i = 1, \ldots, N)$ is a characteristic of spread, e. g. the standard deviation of ξ_x. For the detailed theoretical substantiation of the P-algorithm we refer to (Zhigljavsky and Žilinskas, 2008).

However, the implementation of the original P-algorithm is difficult because the algorithms for $m(x|x_i, y_i, i = 1, \ldots, N)$ and $s(x|x_i, y_i, i = 1, \ldots, N)$ are computationally intensive. To reduce the computational burden the problem is decomposed into a sequential subdivision of $\mathbf{A}$, where $\max \mathbf{P}(\xi_x < y_{on} - \epsilon)$ is evaluated for the hyper-rectangular subsets of $\mathbf{A}$, taking only the function values at the vertices of these hyper-rectangles into account. The algorithm implementing these computational simplifications is presented in (Calvin et al., 2015), where its convergence rate is evaluated as well. This algorithm possesses the first three properties, assumed at the beginning of the section. In the present paper its modification is proposed to improve the performance with respect to the fourth property.

Assuming that the statistical model of ξ_x is Gaussian, Calvin et al. (2015) propose a criterion, in some sense equivalent to the maximum probability (1), computed, however, for a single hyper-rectangular subset of $\mathbf{A}$. In other words, a computationally-efficient criterion value is assigned to each rectangle in the current decomposition of $\mathbf{A}$. Given a rectangle R, its volume V_R and the mean of the function values at its vertices L_R, the criterion is computed as follows:

$$\rho(R, \epsilon) = \frac{V_R}{L_R - y_{0N} + \epsilon}\,, \tag{2}$$

here ϵ depending on the smallest rectangle volume v_{min} in the current decomposition

$$\epsilon(v_{min}) = \begin{cases} q \cdot d \left(v_{min} \cdot \ln(1/v_{min})\right)^{2/d}, & 0 < v_{min} \le \frac{1}{2} \\ q \cdot d, & otherwise \end{cases}; \tag{3}$$

$$q = \frac{3 \cdot 2^{2/3} e^{-1}}{2 \ln(2)}\,, \tag{4}$$

is defined to satisfy the conditions needed for the high convergence rate of the algorithm; see (Calvin et al., 2015).

The original algorithm by Calvin et al. (2015) operates iteratively. At first, the rectangles in the current decomposition are ranked according to the value of (2). Then, a single rectangle with the maximum criterion value is bisected into two equal parts. This is done by performing function evaluations in the middle of the edges along its longest

dimension. Then, again the rectangles are ranked, and so on. The process continues until the maximum allowed number of function evaluations is exceeded.

In the next section we reuse this algorithm as a core of a two-stage approach, where the global and standard phases of operation alternate. Specifically, the criterion (Eq. 2-4) and the bisection procedure are reused.

2.2 The two-phase algorithm

The pseudo-code of the proposed algorithm is given in Algorithm 1.

The algorithm operates by alternating between the two phases: standard and global. Initially, the feasible region is partitioned uniformly into equally-sized rectangles, by dividing each dimension into k equal parts. The rectangles are added to D - the list of active rectangles, i. e. those eligible for further partitioning. Throughout the operation of the algorithm the best known function value y_{0N} and its location in the decision space x_{0N} are tracked. Moreover, the volume histograms of the active, as well as the best rectangle volumes are employed. The best rectangles are those that have a vertex where y_{0N} has been achieved. Accordingly, the values of v_{min}, v_{best}, as the smallest and the largest of active rectangle volumes, and v_{best}, as the largest best rectangle volume, are available.

Iterations of the algorithm are represented by the main while loop. They continue, while the number of function evaluations performed N is smaller than a predefined maximum number N_{max} and while there have been new function evaluations during the previous M consecutive iterations. It is heuristically assumed that once the latter condition fails, the algorithm has found the global minimum and will stagnate around it. Each iteration starts calling the procedure $do\_iteration$, that divides one or more rectangles.

In the beginning of each phase, y_{0N} is memorized as s_{best} (current phase start best value). Initially the phase is set to standard. After the call to $do\_iteration$ the sufficient decrease condition

$$y_{0N} \leq s_{best} - 0.01|s_{best}| \tag{5}$$

is tested. If it is true, s_{best} is updated and $boost\_best$ is set to concentrate the search around the new minimum in the next call to $do\_iteration$. The phase is extended for another iteration, if there is a sufficient decrease or if the volume of the smallest active rectangle is not smaller than Δ. Otherwise, this means that the algorithm has approached some local minimum with the precision Δ and it is time to switch to the global phase. In this case s_{best} is updated and all the rectangles produced up to this point are filtered, so that D contains only the large ones, i. e. those with volume not smaller than T_{volume}:

$$T_{volume}(v_{best}, v_{max}) = \min\left(v_{max}, \frac{v_{max}}{2^{\exp(\tau)}}\right), \tau = \log_2 \frac{v_{max}}{v_{best}} + 1. \tag{6}$$

The global phase is intended to look for the global minimum in the large previously unexplored areas. The switch back to the standard phase occurs only after the sufficient decrease condition (5) is satisfied. Then the s_{best} value is set to the minimum y_{0N} and the search is concentrated around it by setting $boost\_best$. If no sufficient decrease has been achieved, the global phase iterations counter i_{global} is incremented and

the new iteration starts. Every g_{period} iterations a chance is given for the smaller inactive rectangles to be divided by activating rectangles not smaller than v_{best}, invoking $do\_iteration$ once, and again filtering the rectangles according to T_{volume}.

The actual division of rectangles happens in the $do\_iteration$ procedure. Initially, $\epsilon(v_{min})$ is computed according to (Eq. 3 and 4). Based on the value of $boost\_best$, either one or more rectangles are bisected, as in the original algorithm (Calvin et al., 2015). If $boost\_best$ is false, a single rectangle R_i with the largest $\rho(R_i, \epsilon)$ value (see Eq. 2) is divided. If $boost\_best$ is set, then a list of rectangle center distances to x_{0N} is composed and sorted (repeating entries are not removed). The 2^d-th (d is the problem dimension) smallest entry is taken as a distance threshold T_{dist}, and all rectangles with the distance not smaller than that are subdivided.

Several of the above-mentioned values are user-defined parameters. The objective function to be minimized is $objective\_function$, which includes the specification of the feasible region and problem dimension d. The maximum number of function evaluations is N_{max}. During initialization the feasible region is divided into k equal parts along each dimension, in order to increase the search globality in the beginning. The parameter controlling the exploration of the surroundings of any local minimum is Δ. When no function evaluations occur during M consecutive iterations, the program execution terminates, as the algorithm is likely to have found the global minimum. After a multiple of g_{period} global phase iterations has been executed, a single equivalent of the standard phase is invoked.

2.3 Function evaluations data structure

As the rectangles are divided by the points on a regular grid, i. e. $x_{ij} \in \{\frac{1}{2^m}, m \in \mathbb{N}\}$, $i = 1, \ldots, N, j = 1, \ldots, d$, and every point might be a vertex of up to 2^d different rectangles, the repeated function evaluations at the same point are very likely. To avoid them, a tree data structure is developed, storing pairs $(x_i, f(x_i)) \in \mathbb{R}^d \times \mathbb{R}$. Once a need to evaluate the function at some point arises, it is looked up in the history. In case an existing entry is found, it is used and the total number of function evaluations N is not incremented.

The data structure employed is a range tree (Bentley, 1979), i. e. a multi-layered self-balancing binary search tree. Each of its levels corresponds to a single dimension of the feasible region. Namely, the first level is an AVL tree (Knuth, 2010), holding the distinct values of the first coordinate of the input points x_i in its nodes. Moreover, each node itself holds a nested AVL tree. A nested, or k-th layer, tree stores the distinct values of the k-th coordinate of the input points with equal first $k-1$ coordinate values and the nodes have nested AVL trees for further dimensions.

3 Experimental comparison

In this section the operation of the proposed algorithm is illustrated experimentally. The comparison to the original (Calvin et al., 2015) algorithm and the well-known global optimization algorithm DIRECT (Web, a), usually used for similar problems,

Algorithm 1 A two-phase global optimization algorithm pseudo-code.

procedure OPTIMIZE($objective\_function, N_{max}, k, \Delta, M, g_{period}$)
  $D \leftarrow$ rectangles, obtained by partitioning the feasible region into k equal parts along each dimension
  $y_{0N}, x_{0N} \leftarrow$ best function value and its location found so far
  $s_{best} \leftarrow y_{0N}$
  $boost\_best \leftarrow False$
  $i_{global} \leftarrow 0$
  $phase \leftarrow STANDARD$
  $N \leftarrow 0$
  while $N < N_{max}$ and there have been new function evaluations in the M previous consecutive iterations **do**
    DO_ITERATION($D, x_{best}, boost\_best, v_{min}$)
    $boost\_best \leftarrow False$
    $v_{best} \leftarrow$ volume of the largest rectangle containing the value y_{0N}
    $sufficient\_decrease \leftarrow$ is y_{0N} better than s_{best} by 1%?
    if $phase == STANDARD$ **then**
      if $sufficient\_decrease$ **then**
        $s_{best} \leftarrow y_{0N}$
        $boost\_best \leftarrow True$
      else if $v_{min} < \Delta$ **then**
        $s_{best} \leftarrow y_{0N}$
        $phase \leftarrow GLOBAL$
        $D \leftarrow$ rectangles that are not smaller than $T_{volume}(v_{best}, v_{max})$
      end if
    else
      $i_{global} \leftarrow i_{global} + 1$
      if $sufficient\_decrease$ **then**
        $s_{best} \leftarrow y_{0N}$
        $boost\_best \leftarrow True$
        $i_{global} \leftarrow 0$
        $phase \leftarrow STANDARD$
        $D \leftarrow$ rectangles that are not smaller than v_{best}
      else if $i_{global}\ mod\ g_{period} == 0$ **then**
        $D \leftarrow$ rectangles that are not smaller than v_{best}
        DO_ITERATION($D, x_{0N}, boost\_best, v_{min}$)
        $D \leftarrow$ rectangles that are not smaller than $T_{volume}(v_{best}, v_{max})$
      end if
    end if
  end while
end procedure
procedure DO_ITERATION($D, x_{0N}, boost\_best, v_{min}$)
  $\epsilon \leftarrow \epsilon(v_{min})$
  if $boost\_best$ **then**
    $T_{dist} \leftarrow$ largest of the 2^d smallest distances of rectangles from x_{0N}
    Divide rectangles with the distance to x_{0N} not less than $thres$
  else
    Divide the rectangle R_i, for which $rho(R_i, \epsilon)$ is maximum
  end if
  Update $N, y_{0N}, x_{0N}, v_{min}, v_{max}, v_{best}$
end procedure

Table 1. Testing results with functions from Hansen and Jaumard (1995) defined over a 2-dimensional region. Optimization stops after a predefined number of function evaluations N.

Alg.	*Proposed*			*Original*			*DIRECT*		
	N			N			N		
func.	500	1000	3000	500	1000	3000	500	1000	3000
1	5.30	5.30	5.80	4.30	4.30	5.30	4.14	4.14	∞
2	6.93	6.93	6.93	5.65	5.65	6.93	3.71	3.71	3.71
3	6.77	6.77	8.18	12.48	13.47	13.47	5.58	5.58	5.58
3.1	6.75	6.75	7.21	12.29	13.39	13.47	5.58	5.58	5.58
3.2	6.75	6.75	7.21	12.29	13.39	13.47	5.58	5.58	5.58
3.3	8.07	8.07	8.18	13.34	13.34	13.34	5.58	5.58	5.58
4	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	3.97	4.14	6.01	3.12	5.33	5.33	4.48	4.48	4.48
6	2.66	4.49	4.49	3.87	3.87	3.87	1.78	1.78	1.78
7	2.15	4.02	9.42	∞	∞	∞	2.87	3.74	6.67
8	6.12	6.12	6.23	4.91	4.91	6.12	4.30	4.70	4.70
9	∞	∞	∞	-1.43	-1.43	-1.43	∞	∞	∞
9.1	∞	∞	∞	-1.43	∞	∞	∞	∞	∞
9.2	6.40	9.41	12.40	∞	∞	∞	∞	∞	∞
9.3	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	9.17	9.22	11.23	13.40	13.78	13.78	4.64	4.64	4.64
11	7.03	9.17	9.17	5.62	7.44	12.36	∞	∞	∞
12	4.65	6.02	9.04	4.23	5.16	7.84	4.32	4.32	4.32
13	1.20	1.51	1.56	0.99	1.02	1.29	1.29	1.29	1.86

is included. Throughout the section, the three algorithms are denoted by "Proposed", "Original" and "DIRECT", respectively.

Two types of experiments have been carried out. First, a number of 2-dimensional testing functions from (Hansen and Jaumard, 1995) have been minimized. The results are presented in Table 1. The first column contains the IDs of the objective functions, as they appear in the original source. For each algorithm, the obtained precision e_N after $N = 500, 1000, 3000$ function evaluations is given, where $e_N = -log_{10}(f^* - y_{0N})$, f^* is the global minimum and y_{0N} is the best value found. The larger e_N, the higher precision has been achieved. The symbol ∞ means that the exact global minimum has been found. The results for DIRECT have been obtained with an implementation from (Web, a). The following values of the parameters of the proposed algorithm have been used: $\Delta = 10^{-9}$, $M = 5$, $k = 4$, $g_{period} = 20$.

It can be seen that the Proposed and Original algorithms perform similarly on the first set of testing functions, and DIRECT performs slightly worse. The Original algorithm fails in approximating the global minimum of the 9-th function. All algorithms find it difficult to approximate the 13-th objective function, due to its global minimizer being hidden behind areas of high function levels. Also, DIRECT found a relatively rough solution to the 6-th problem. Apart from that, all algorithms manage to find the global minima with acceptable precision.

For the second experiment, the free implementation (Web, b) of the GKLS test function generator (Gaviano et al., 2003)) was used. Its authors emphasize the need

to have the testing functions of controllable difficulty with known properties, such as the number and locations of local minima. It is possible to generate non-differentiable, continuously differentiable or twice continuously differentiable testing functions. The functions are obtained by distorting a quadratic function by polynomials. There are several parameters, that are needed to specify a completely reproducible function class: d - the problem dimension, n_{local} - the number of local minima, f^* - the global minimum value, ρ^* - the radius of the attraction region of the global minimizer, r^* - the distance between the global minimizer and the quadratic function vertex. Problem difficulty increases with d, n_{local} and r^*. Each class contains 100 different testing functions.

We based our experiment on the methodology presented in Sergeyev and Kvasov (2006) and Paulavičius et. al (2014). In our experiment we used the first 6 continuously differentiable function classes with dimensionality $d = 2, 3$ and 4 from the aforementioned sources. The parameters common to all classes are $f^* = -1$ and $n_{local} = 10$. The rest of the class parameters are specified in the results tables (Tables 2 and 3) next to the class number.

The stopping condition for all algorithms was:

$$\exists i \in \{1, \ldots, N\} : |x_{ij} - x_j^*| \leq \sqrt[d]{\Delta}|b_j - a_j|, \tag{7}$$

where $A = \{t : \mathbb{R}^d : a_j \leq t_j \leq b_j, j = 1, \ldots, d\}$ is a hyper-rectangular feasible region. This means that the algorithm generated a point x_i such that the area of a rectangle with x_i and the global minimizer x^* as its opposite vertices has the volume not larger than Δ times the volume of the feasible region. In case an algorithm does not satisfy this condition in $N_{max} = 1000000$ function evaluations, its is stopped as well.

The performance of the algorithms has been assessed based on the number of function evaluations needed to reach the stopping condition (7) for the 100 functions in each class. Two criteria are used:

1. The maximum number of function evaluations N_k performed for the class:

$$\max_{k=1,\ldots,100} N_k. \tag{8}$$

2. The average number of function evaluations performed for the class:

$$\frac{1}{100} \sum_{k=1}^{100} N_k \, . \tag{9}$$

The parameters of the proposed algorithm were set as follows: $M = \infty$, $k = 4$, $g_{period} = 20$. The value of Δ corresponds to the condition (7) and is given in Tables 2 and 3 for each function class. The results for DIRECT were taken from Sergeyev and Kvasov (2006).

Table 2 shows the first criterion values for the compared algorithms. Each row represents a single function class. For each of the algorithms, the 100 functions were sorted in an ascending order according to the number of function evaluations required to satisfy the condition (7). The maximum among the first half of entries is given in the columns under the "50%" heading. The values of (8) are listed under the "100%" heading. The

best value in 50% and 100% categories is highlighted. The fact that an algorithm exceeded $N_{max} = 1000000$ function evaluations, is denoted by $> 1000000(k)$, where k is the number of functions, for which this happened. The second criterion values are listed in Table 3. The class parameters are given analogously to Table 2.

According to the first criterion, the Proposed algorithm significantly improves upon the Original algorithm. It also performs better than DIRECT, which does not manage to satisfy condition (7) for all the functions in the classes $4-6$. This happens for the Original algorithm as well, with 3 functions from the last class. The second criterion shows that the Proposed algorithm is the best for the harder classes in the two and three-dimensional cases, also for classes 5 and 6. It seems that DIRECT performs better on average, but is less prone to the worst-case functions. The proposed two-phase algorithm seems to effectively extend the Original algorithm, protecting against the problems arising with harder-than-average functions.

Table 2. Results with GKLS-generated function classes. The maximum number of function evaluations for each class.

Class	d	Δ	r^*	ρ^*	50%			100%		
					Proposed	*Original*	*DIRECT*	*Proposed*	*Original*	*DIRECT*
1	2	10^{-4}	.90	.2	210	382	**111**	**424**	1992	1159
2	2	10^{-4}	.90	.1	**605**	1638	1062	**1295**	13848	3201
3	3	10^{-6}	.66	.2	2705	10117	**386**	**5623**	34009	12507
4	3	10^{-6}	.90	.2	3446	14281	**1749**	**8333**	131325	>1000000(4)
5	4	10^{-6}	.66	.2	10869	52593	**4805**	**41151**	569193	>1000000(4)
6	4	10^{-6}	.90	.2	**13604**	95313	16114	**47532**	>1000000(3)	>1000000(7)

Table 3. Results with GKLS-generated function classes. The average number of function evaluations for each class.

Class	d	Δ	r^*	ρ^*	100%		
					Proposed	*Original*	*DIRECT*
1	2	10^{-4}	.90	.2	217.76	477.22	**198.89**
2	2	10^{-4}	.90	.1	**602.5**	2340.46	1063.78
3	3	10^{-6}	.66	.2	2777.38	10667.24	**1117.70**
4	3	10^{-6}	.90	.2	**3814.69**	21512.76	>42322.65
5	4	10^{-6}	.66	.2	**13208.57**	77987.08	>47282.89
6	4	10^{-6}	.90	.2	**18098.15**	>161152.25	>95708.25

4 Conclusions

An algorithm, based on the ideas of statistical modeling of the black-box objective function, is extended to balance the global and local phases of the search for the global

minimum. The proposed algorithm has been tested with over 600 objective functions and compared to the original (Calvin et al., 2015) and the well-known DIRECT (Web, a) algorithms. The results show that the extension is beneficial, when the functions are hard, that is multimodal, having relatively small regions of attraction of the global minimum, whose location is not quickly deducible from the information gathered about the function. With this extension the algorithm becomes comparable to and better than DIRECT. This means that it is worthwhile seeking further computational reductions by balancing the global and local search phases in the statistical global optimization.

Acknowledgements

This work was supported by the Research Council of Lithuania under Grant No. MIP-051/2014.

References

Bentley, J. L. (1979). Decomposable Search Problems. Information Processing Letters. Vol. 8(5), 244-251.

Calvin, J. M., Žilinskas, A. (2014). On a Global Optimization Algorithm for Bivariate Smooth Functions. Journal of Optimization Theory and Applications. Vol. 163(2), 528-547.

Calvin, J. M., Gimbutienė, G., Phillips, W. O., Žilinskas, A. (2015). On a Global Optimization Algorithm for Multivariate Smooth Functions. Submitted.

Fishburn, P. (1970). Utility Theory for Decision Making. Wiley.

Floudas, C. (2000). Deterministic Global Optimization: Theory, Algorithms and Applications. Kluwer Academic Publishers.

Gaviano, M., Lera, D., Kvasov, D. E., Sergeyev, Y. D. (2003). Algorithm 829: Software for Generation of Classes of Test Functions with Known Local and Global Minima for Global Optimization. ACM Transactions on Mathematical Software (TOMS). Vol. 29(4), 469-480.

Hansen, J., Jaumard, B. Lipschitz Optimization (1995). In: Horst, R., Pardalos, P. (Eds.), Handbook of Global Optimization, Kluwer Academic Publishers, Dodrecht, 407-493.

Horst, R., Pardalos, A. P., Thoai, N. (1995). Introduction to Global Optimization. Kluwer Academic Publishers.

Knuth, D. E. (2010). The Art of Computer Programming. 3rd ed., vol. 3, Addison-Wesley, Upper Saddle River.

Mockus, J. (1989). Bayesian Approach to Global Optimization. Kluwer Academic Publishers, Dodrecht.

Pardalos, P., Romeijn, H. (2002). Handbook of Global Optimization. Vol. 2. Springer.

Paulavičius, R., Sergeyev, Y. D., Kvasov, D. E., Žilinskas, J. (2014). Globally-biased DISIMPL algorithm for expensive global optimization. Journal of Global Optimization 59.2-3, 545-567.

Paulavičius, R., Žilinskas, J. (2014). Simplicial Global Optimization. Springer, New York.

Pinter, J. (1996). Global Optimization in Action. Kluwer Academic Publishers.

Sergeyev, Y. D., Kvasov, D. E. (2006). Global Search Based on Efficient Diagonal Partitions and a Set of Lipschitz Constants. SIAM Journal on Optimization 16.3, 910-937.

Sergeyev, Y. D., Kvasov, D. E. (2011). Lipschitz Global Optimization. In: J. Cochran, L. Cox, P. Keskinocak, J. Kharoufeh, J. Smith (Eds.), Wiley Encyclopedia of Operations Research and Management Science, Kluwer Academic Publishers, Dodrecht, Vol. 4, 2812-2828.

Strongin, R., Sergeyev, Y. (2000). Global Optimization with Non-convex Constraints: Sequential and Parallel Algorithms. Kluwer Academic Publishers.
Zhigljavsy, A., Žilinskas, A. (2008). Stochastic Global Optimization. Springer.
Žilinskas, A., Žilinskas J. (2002). Global Optimization Based on a Statistical Model and Simplicial Partitioning. Computers & Mathematics with Applications. Vol. 44(7), 957-967.
Web (a). DIRECT - A Global Optimization Algorithm. http://www4.ncsu.edu/~ctk/Finkel_Direct/
Web (b). GKLS test functions generator implementation home page. http://wwwinfo.deis.unical.it/~yaro/GKLS.html

Received August 15, 2015 , accepted August 31, 2015