DOI: 10.14529/cmse230301

ON USING THE DECISION TREES TO IDENTIFY THE LOCAL EXTREMA IN PARALLEL GLOBAL OPTIMIZATION ALGORITHM*

© 2023 K.A. Barkalov, I.G. Lebedev, D.I. Silenko

Lobachevsky State University of Nizhny Novgorod
(pr. Gagarina 23, Nizhny Novgorod, 603022 Russia)
E-mail: barkalov@vmk.unn.ru, ilya.lebedev@itmm.unn.ru, silenko@itmm.unn.ru

Received: 31.07.2023

In the present work, the solving of the multidimensional global optimization problems using decision tree to reveal the attractor regions of the local minima is considered. The objective function of the problem is defined as a "black box", may be non-differentiable, multi-extremal and computational costly. We assume that the function satisfies the Lipschitz condition with a priory unknown constant. Global search algorithm is applied for the search of global minimum in the problems of such type. It is well known that the solution complexity essentially depends on the presence of multiple local extrema. Within the framework of the global search algorithm, we propose a method for selecting the vicinity of local extrema of the objective function based on analysis of accumulated search information. Conducting such an analysis using machine learning techniques allows making a decision to run a local method, which can speed up the convergence of the algorithm. This suggestion was confirmed by the results of numerical experiments demonstrating the speedup when solving a series of test problems.

Keywords: global optimization, multiextremal functions, parallel computing, machine learning, decision tree.

FOR CITATION

Barkalov K.A., Lebedev I.G., Silenko D.I. On Using the Decision Trees to Identify the Local Extrema in Parallel Global Optimization Algorithm. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 3. P. 5–18. DOI: 10.14529/cmse230301.

Introduction

In the present work, we consider the parallel algorithms for solving the multidimensional global optimization problems. Such problems often arise in the cases, when it is necessary to select the values of parameters of the mathematical model being investigated, in which the simulation results best fit the experimental data. For solving the problems of this class, many algorithms are known: from the metaheuristic algorithms, based on the idea of random search [1–3], to the deterministic algorithms guarantying convergence to the global minimum [4–6].

Since in real global optimization problems each computing of the function value (hereafter called *a trial*) is a computation-costly operation, one has to reduce the number of such trials. It can be achieved by intentional choice of variants in the course of search for the optimal solution cutting off unpromising search subdomains and investigating only the ones, in which the solution of the problem can be found. The global search algorithm (GSA) is based on this idea [7]. In the present work, we tried to combine GSA and a local optimization method (Hooke–Jeeves pattern search method [8]) to reduce the number of trials executed. The decision on the run of the local method will be made using a decision tree.

*The paper is recommended for publication by the Program Committee of the International Scientific Conference "Parallel Computational Technologies (PCT) 2023".

The local optimization algorithms are intended for determining only one of local extrema within a set of feasible solutions, in which the objective function takes the extremal (maximum or minimum) value. Among the local methods, the zero-order methods and the gradient methods (of the 1st order and of the 2nd one) are traditionally distinguished [9]. Note that the application of the first-order methods (not to mention the second one) in the problems with the "black-box" type functions is difficult since the problem of numerical estimating the gradient arises here [10]. Therefore, further we will consider only the zero-order methods.

In order to determine at which moment it is best to run the local method within the framework of the global search, we will use an algorithm based on a decision tree. At the first search stage, the search information is accumulated in the form of the results of the trials executed. We used the trial results for training the decision tree that allows obtaining a piecewise constant approximation of the objective function and predicting the objective function behavior on its base. To do so, we will compare the function values at the points neighboring to the one, which we consider to be suspicious for a local minimum, and make a decision whether the next trial point falls into the attractor region of a local minimum or not.

The main part of the paper has the following structure. In Section 1, the global optimization problem statement is considered, the basic suggestions on the objective function are made. In Section 2, the description of the methods and approaches used is given. In particular, the main idea of global search algorithm is discussed and its computational rules are given. The section also contains the description of the local search method used and the method of constructing the approximation of the function using the decision trees. The novel algorithm combining the global search and the local one on the base of using the decision trees is presented in Section 3. In the next section, the features of the asynchronous parallelization scheme of the new algorithm are discussed. Section 5 contains the results of experiments, carried out using a parallel computer system with distributed memory, and a brief conclusion.

1. Problem statement

Let us consider a problem of searching the global minimum of a function $\varphi(y)$ in a hyperinterval $D = \{y \in R^N : a_i \leq y_i \leq b_i,\ 1 \leq i \leq n\}$.

$$\varphi(y^*) = min\{\varphi(y) : y \in D\}, D = \{y \in R^N : a_i \leq y_i \leq b_i, 1 \leq i \leq N\}, \quad (1)$$

where $a, b \in R$ are given vectors.

Also, assume the function to satisfy the Lipschitz condition with *a priori* unknown constant L that corresponds to limited variation of the function values at limited variation of the argument.

$$|\varphi(y_1) - \varphi(y_2)| \leq L \parallel y_1 - y_2 \parallel, y_1, y_2 \in D, 0 < L < \infty.$$

This suggestion can be considered (with respect to the real-world problems) as the reflection of limited power generating the changes in the system being simulated.

The numerical solving of the problem (1) is reduced to constructing an estimate $y_k^* \in D$ matching some concept of nearness to the point y^* (for example, $||y^* - y_k^*|| \leq \varepsilon$, where $\varepsilon \geq 0$ is a predefined accuracy) based on a finite number k of the objective function values computed. With respect to the class of the considered problems, it is suggested that the objective function $\varphi(y)$ can be defined algorithmically, as a result of executing some subroutine or library.

The solving of the global optimization problems is much more computationally expensive as compared to the local optimization problems since finding the global optimum requires ex-

ploration of the whole search domain. As a result, the search of the global optimum is reduced to constructing some coverage in the search domain and choosing the best function value on this grid. When using the uniform grids, the computation costs of solving the problem grow exponentially with increasing dimensionality.

2. Algorithms

The main idea of the approach to constructing more efficient nonuniform grid is that its points are computed sequentially while the objective function is considered as a realization of some random process. The decision rules of the grid building algorithm are constructed in such a way that the next grid point corresponds to the global minimum point of the mathematical expectation of the function values. This point is stored in the list of known values, and the iterations are repeated until the stop criteria is satisfied: either the distance between the trial points becomes less than a predefined value or a preset maximum number of iterations is achieved.

When solving the multidimensional problems, the dimensionality reduction (i.e. the reduction of the multidimensional problem to an equivalent one-dimensional one) using Peano curves is applied. These ones allow reducing a multidimensional optimization problem in the domain D to a one-dimensional minimization problem within the interval $[0,1]$

$$\varphi(y(x^*)) \ = \min\{\varphi(y(x)) : x \in [0,1]\},$$

where the function $\varphi(y(x^*))$ satisfies more general Hölder condition

$$|\varphi(y\,(x_1)) - \varphi(y\,(x_2))\,| \leq \ H\,|x_1 - x_2|^{\frac{1}{N}}\,,\ x_1,\ x_2 \epsilon [0,1].$$

Therefore, instead of the initial problem of minimizing the function $\varphi(y)$ in the domain D, we can consider the minimization of the one-dimensional function $f(x) = \varphi(y(x))$ satisfying the Hölder condition for $x \in [0,1]$.

2.1. Multidimensional parallel global search algorithm

The main steps of the parallel global search algorithm are as follows.

At the preliminary step, p trials are executed in parallel in arbitrary internal points $x^1, ..., x^p$ of the interval $[0,1]$ that corresponds to the first iteration of the algorithm.

If $n \geq 1$ iterations are completed, which correspond to $k = k(n)$ trials executed at the points x^i, $1 \leq i \leq k$, then the points $x^{k+1}, \ldots, x^{k+p}$ of the search trials at the next $(n+1)^{\text{th}}$ iteration will be computed as the result of performing the following operations.

1. Renumber (by the lower indices) the points x^i, $1 \leq i \leq k$ as well as the boundary points of the interval $[0,1]$ in increasing order of the coordinate

$$0 = x_0 < \ x_1 < \ \ldots \ < x_{k+1} = 1. \tag{2}$$

   and juxtapose them with the values $z_i = f(x_i)$.
2. Compute current lower estimate M of the unknown Hölder constant H:

$$\mu = max\left\{\frac{|z_i - z_{i-1}|}{(x_i - x_{i-1})^{1/N}},\ i = 1, \ldots, k\right\},\ M = \begin{cases} r\mu,\ \mu > 0, \\ 1,\ \mu = 0, \end{cases} \tag{3}$$

where $r > 1$ is a parameter of algorithm. This parameter controls the reliability of the algorithm: the higher values of r ensure guaranteed finding of the global minimum, the choice of lower value speeds up the convergence of the algorithm.

3. For each interval (x_{i-1}, x_i), $1 \le i \le k+1$, compute the value $R(i)$ called *a characteristic* of the interval according to the formulae

$$R(1) = 2\Delta_1 - 4\frac{z_1}{M},\ R(k+1) = 2\Delta_{k+1} - 4\frac{z_k}{M}, \tag{4}$$

$$R(i) = \Delta_i + \frac{(z_i - z_{i-1})^2}{M^2\Delta_i} - 2\frac{z_i + z_{i-1}}{M}, 1 < i < k+1, \tag{5}$$

where $\Delta_i = (x_i - x_{i-1})^{\frac{1}{N}}$.

4. Arrange the characteristics $R(i)$, $1 \le i \le k+1$, in the non-increasing order

$$R(t_1) \ge\ R(t_2) \ge ... \ge\ R(t_k) \ge\ R(t_{k+1}), \tag{6}$$

and select p intervals with the indices t_j, $1 \le\ j \le\ p$ with the largest values of characteristics.

5. Compute the points $x^{k+j},\ 1 \le j \le p$ in the selected intervals according to the formulae

$$x^{k+j} = \frac{x_{t_j} + x_{t_j-1}}{2},\ t_j = 1,\ t_j = k+1, \tag{7}$$

$$x^{k+1} = \frac{x_{t_j} + x_{t_j-1}}{2} - sign\left(z_{t_j} - z_{t_j-1}\right)\frac{1}{2r}\left[\frac{\left|z_{t_j} - z_{t_j-1}\right|}{\mu}\right]^N,\ 1 < t_j < k+1. \tag{8}$$

The next p trials are executed in parallel at the points $x^{k+j},\ 1 \le j \le p$, computed according to the formulae (7), (8). Upon completing the trials, the results of these ones are stored in the information database, and the algorithm goes to computing new trial points. Note that as a rule the process of executing a trial in the applied optimization problems is much more computation-costly as compared to computing the trial point.

The algorithm stops in the case if the condition $\Delta_{t_j} < \varepsilon$ is satisfied for even a single value t_j, $1 \le\ j \le\ p$, from (6). This stop criterion (along with the criterion limiting the number of executed iterations usual for the iteration methods) is used in the applied optimization problems, in which the global minimum point y^* is unknown a priori.

When solving the test problems, in which the global minimum point y^* is known, we can use the stop criterion upon falling into the vicinity of the global minimum as well. In this case, the method stops if the condition $\left\|y(x_{t_j}) - \ y^*\right\| < \varepsilon$ is satisfied even for one value of $t_j,\ 1 \le\ j \le\ p$, from (6).

As the final estimate of the global optimizer of the considered problem, the values

$$f_k^* = \min_{1 \le i \le k} f(x_i),\ x_k^* = arg\min_{1 \le i \le k} f(x_i). \tag{9}$$

are taken.

The substantiation of this method of organization of computations see in [7, 11]. The modifications taking into account the presence of inequality constraints as well as the information on the derivative of the objective function are presented in [12, 13].

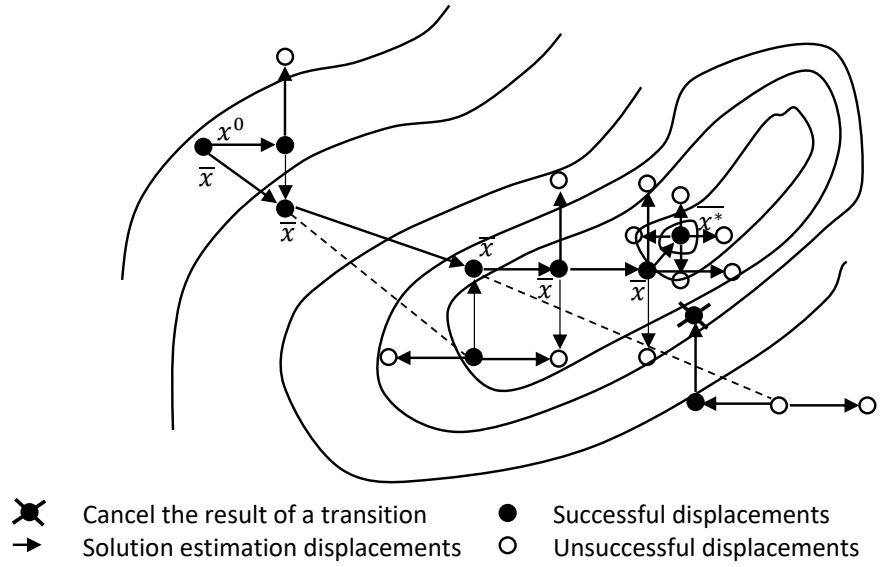

Figure 1. Example of iterations of the Hooke–Jeeves algorithm

2.2. Hooke–Jeeves method

For the Hooke–Jeeves pattern search algorithm belongs to the class of zero-order methods. Its computation rules are a combination of the investigating search (to select the direction) and search in the direction selected [14].

The investigating search is performed as follows.

1. The step value is determined (it is different for each coordinate and can be varied in the course of search).
2. The search step is considered to be successful if the value of the objective function at the check point does not exceed the value of the objective function at the initial point.
3. Otherwise, it is necessary to return to the previous item and make a step in the reverse direction.
4. After making the steps in all N coordinates, the investigating search is completed. The point obtained is called the base point.

Figure 1 shows the example of the algorithm work. Function level lines are displayed, filled circles indicate successful steps, unfilled circles correspond to unsuccessful steps performed during the investigating search. With regard to the search in a selected direction, it consists in performing a step from the base point found during the investigating search along the line connecting this one with the previous base point. The magnitude of this step is defined by a parameter set in advance.

2.3. Decision tree

The decision tree is a tool used for an automated analysis of big data arrays, which is applied in the machine learning. Decision tree is a binary tree (a tree, in which each non-leaf node has two child nodes). It can be used for solving the classification problems as well as the regression ones. When solving the classification problems, each leaf of the tree is marked by a class label, and several leafs can have the same labels. In the case of constructing a regression, a constant is assigned to each leaf of the tree. Therefore, the approximating function obtained is a piecewise-

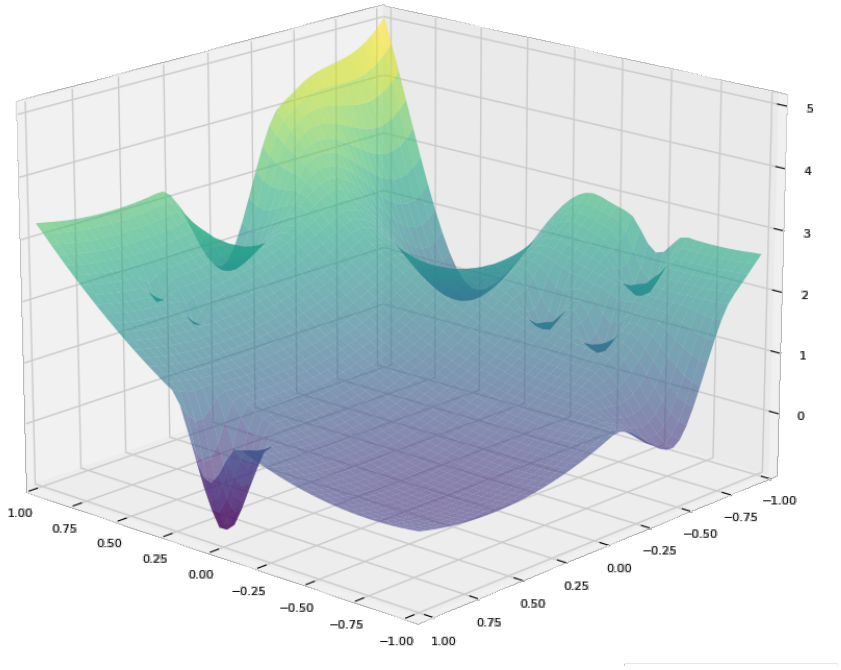

Figure 2. Objective function $\varphi(y)$ from the test class

constant one. In our case, the decision tree constructs the function $\psi(y)$ as a piecewise-constant approximation of $\varphi(y)$ in the search domain. Let us denote the value computed by the decision tree at the point y as $z' = \psi(y)$. Figure 2 presents a graph of the objective function $\varphi(y)$; Fig. 3 presents corresponding approximation $\psi(y)$ built using decision tree.

When implementing the algorithm for finding the attraction areas of the local minima, we used the algorithms from OpenCV library to construct decision tree. OpenCV is an open source library of the algorithms of computer vision, image processing, and general-purpose numerical algorithms. More details on the decision tree can be found in [15].

3. Combination of local and global search algorithms for solving multidimensional problems

In the current section, let us present a detailed description of how we used the decision tree for finding the attraction areas of the local extrema. In the course of running GSA, it is necessary to determine whether it is worth to use current point as a start one for the local method or not. To do so, one can check the points of adjacent trials. If among these ones there are no points at which the function values are lower than in current one, one can suggest that we are in the attractor of a local minimum. In this case, we can run the local method from current point, which will rapidly converge to a local minimum of the function. We can do it in the multidimensional space only, not in one-dimensional interval after the dimensionality reduction. First, the reduced function $\varphi(y(x))$ changes its properties: one local minimum in the multidimensional space may divide into a set of minima after the dimensionality reduction. Second, after the use of the mapping $y(x)$, we can lose the information on the mutual arrangement of the points in the original space. The points located close to each other in the multidimensional space may appear to be essentially separated in the one-dimensional interval.

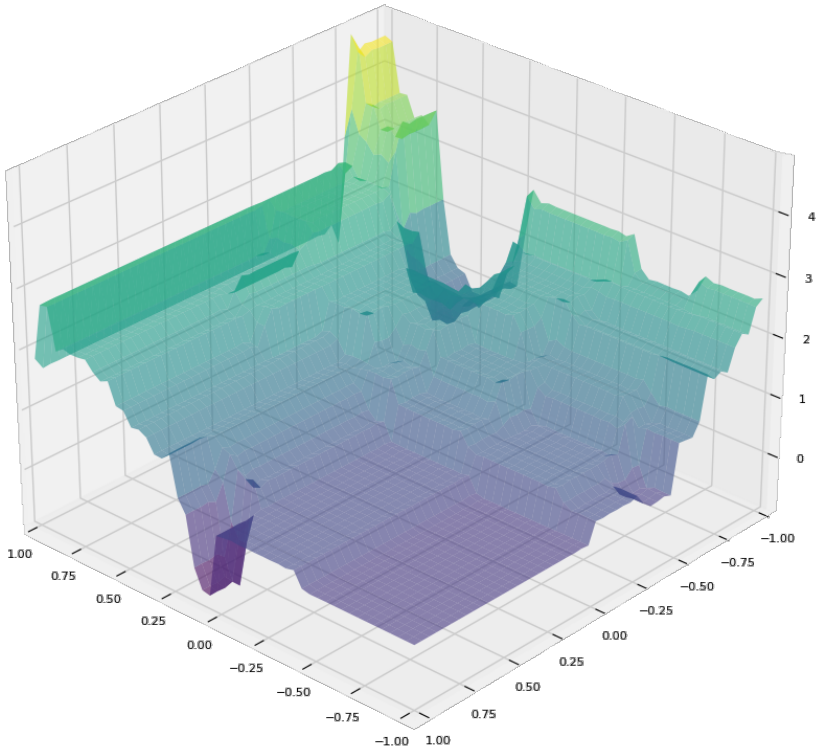

Figure 3. Piecewise-constant approximation $\psi(y)$ of the function $\varphi(y)$

Therefore, we will determine the attraction areas of the local extrema in the original multidimensional space. However, the determining of the adjacent points in the multidimensional space is quite expensive in terms of computing resources. To do so, we will use the decision tree.

After completing a certain number of trials (for example, $100 \cdot N$), let us use all accumulated points to initialize a suitable data structure and training the decision tree on its base. To make the determining of the adjacent points easier, we construct a uniform grid with certain step in each coordinate. Next, we compute the values of the approximation of the objective function in these points using the decision tree. Now, having a uniform grid of points and knowing the function values in each point, let us find the closest point to the initial one (from the viewpoint of the Euclidian distance). This closest point is a projection of the initial point onto the uniform grid, on which its neighbors can be determined easily. Since the decision tree constructs the approximation of the initial function (even piecewise constant), we can estimate the function values in the areas, in which the trials have not been executed earlier.

When considering the neighbors, it is necessary to take into account the following. If even a single neighbor has a lower function value than current point, there is no need to run the local method. If at even a single neighboring point the function value is the same as at the current one, this point must be also checked (i.e., to find all its neighbors and to check these ones in the same way). Only in the case when all function values at the neighboring points are greater than at the current point, we can run the local method.

4. Asynchronous algorithm parallelization scheme

The method of parallel computing described in previous section implies synchronous performing of p search trials at the points computed at current iteration. The next iteration will be started only upon completing all trials. In the case of different computation costs of executing the trials at different points of the search domain, this approach may lead to a disbalance of the computing jobs. This drawback can be corrected by introduction of asynchrony. The idea of the asynchronous scheme is not to wait for completing all p trials but to initiate the execution of a new trial immediately upon completing one of the trials.

Let us assume that the master process computes one point of the next trial at every iteration and sends it to the slave process to execute the trial. Note that executing a trial by the slave process in the applied optimization problems is much more computation costly than the choice of a new trial point by the master process. Therefore, the idle time of the processes will be negligible. In this case (unlike the synchronous parallel algorithms) total number of trials executed by each slave process will depend on the computation costs for executing particular trials and cannot be estimated in advance. When describing the parallel algorithm, let us assume that we have one master and p slave processes at our disposal.

At the initial phase the master process (assume it to have the identifier 0) starts p trials in parallel at p different points $\{y\left(x^1\right), y\left(x^2\right), \ldots, y\left(x^p\right)\}$. The inverse images of two points are the boundary ones $x^1 = 0$, $x^p = 1$; the rest are the internal points $x^i \in (0,1)$, $i = 2, \ldots, p-1$.

Now assume $k \geq 0$ trials to be completed, and the slave processes are performing the trials at the points $y\left(x^{k+1}\right)$, $y\left(x^{k+2}\right), \ldots,$ $y\left(x^{k+p}\right)$.

Each slave process having completed the computing of the function at some time moment sends the computed value to the master process. Next, the master process selects a new point x^{k+p+1} for the slave process according to the rules provided below. Note that in this case it will be a set of inverse images of the trial points $I_k = \left\{x^{k+1}, x^{k+2}, \ldots, x^{k+p}\right\}$, in which the trials have been already initiated but have not been completed yet.

So far, the parallel asynchronous global optimization algorithm using decision tree to find the local extrema attractors consists of the following steps.

1. Arrange in the increasing order (by the lower indices) the set of inverse images of the trial points

$$X_k = \left\{x^1, x^2, \ldots, x^{k+p}\right\},$$

containing all points, in which the trials either have been completed or are being executed, i.e. get an ordered set

$$0 = x_1 < x_2 < \ldots < x_{k+p} = 1.$$

2. Compute the values

$$M_1 = \max\left\{\frac{|z_i - z_{i-1}|}{(x_i - x_{i-1})^{1/N}} : x_{i-1} \notin I_k, x_i \notin I_k, 2 \leq i \leq k+p\right\},$$

$$M_2 = \max\left\{\frac{|z_{i+1} - z_{i-1}|}{(x_{i+1} - x_{i-1})^{1/N}} : x_i \in I_k, 2 \leq i < k+p\right\},$$

$$M = \max\{M_1, M_2\},$$

where $z_i = \varphi(y(x_i))$ if $x_i \notin I_k,\ 1 \le i \le k+p$. The values z_i at the points $x_i \in I_k$ are not defined as far as the trials at these points have not been completed yet. If $M = 0$, then set $M = 1$.

3. Juxtapose each interval $(x_{i-1}, x_i),\ x_{i-1} \notin I_k, x_i \notin I_k,\ 2 \le i \le k+p$, with its characteristic $R(i)$ computed according to the formula

$$R(i) = rM\Delta_i + \frac{(z_i - z_{i-1})^2}{rM\Delta_i} - 2(z_i + z_{i-1}),$$

where $\Delta_i = (x_i - x_{i-1})^{1/N}$ and $r > 1$ is the reliability parameter of the method.

4. Find the interval $[x_{t-1}, x_t]$, which corresponds to the highest characteristic, i.e.

$$R(t) = \max\{R(i) :\ x_{i-1} \notin I_k, x_i \notin I_k,\ 2 \le i \le k+p\}.$$

5. Compute a new trial point $y^{k+p+1} = y\left(x^{k+p+1}\right)$, the inverse image of which is $x^{k+p+1} \in (x_{t-1}, x_t)$, according to the formula

$$x^{k+p+1} = \frac{x_t + x_{t-1}}{2} - \operatorname{sign}(z_t - z_{t-1})\frac{1}{2r}\left[\frac{|z_t - z_{t-1}|}{M}\right]^N.$$

6. Get the computed function value from the process j, add new trial $z_j = f(y(x_j))$ to the set V.
7. If $k < 100N$, return to Step 1.
8. Create the decision tree using the set I_k, get the approximating function $\psi(y)$.
9. If the decision tree is used for the first time, construct a uniform grid

$$Y' = \{y' \in R^N :\ a_i \le y'_i \le b_i,\ 1 \le i \le N\},$$

where number of the grid nodes is a parameter of the method.

10. Compute the approximation values: $Z' = \{z' = \psi(y'), y' \in Y'\}$
11. For all points $y' \in V$:
    Find the points y'_q closest to y',
    Make a detour of the neighbors y'_q according to the principle described above.
    If not one neighbor point has a lower value than in y'_q, run the local method.
    Clear the set V.

After the next trial point is computed, the master process includes it to the set I_k and sends to the slave process, which starts the new trial at this point. The master process stops the algorithm, if at least one of the two conditions is satisfied: $\Delta_t <$ or $k + p > K_{max}$. The real value $\epsilon > 0$ and the integer value $K_{max} > 0$ are the algorithm parameters. They correspond to the accuracy of the search and the maximum number of trials, respectively.

5. Numerical experiments

The numerical experiments were carried out on Lobachevsky supercomputer. Each supercomputer node had two processors Intel Sandy Bridge E5-2660 2.2 GHz, 64 Gb RAM.

In the experiments, we used GKLS generator of test problems, which can generate multiextremal optimization problems with known properties: the global minimum point, the number of local minima, etc.

To demonstrate the efficiency of investigated global search algorithm, here we present the results of comparing this one with well known methods DIRECT and DIRECTl. Table 1 presents the averaged numbers of iterations executed by respective methods when solving a series of GKLS problems. The sign " $>$ " points to the situation when not all problems are solved. In this case, the number of non-solved problems is given in brackets. As one can see, GSA overcomes DIRECT and DIRECTl methods in average number of iterations necessary for solving the problems with the same precision.

Table 1. Averaged number of iterations

N	Problem class	DIRECT	DIRECTl	GSA
4	Simple	$>$ 47 282 (4)	18 983	11 953
	Hard	$>$ 95 708 (7)	68 754	25 263
5	Simple	$>$ 16 057 (1)	16 758	15 920
	Hard	$>$ 217 215 (16)	$>$ 269 064 (4)	$>$ 148 342 (4)

Below, the results of comparing two parallel algorithms — asynchronous global search algorithm (AGSA) and its modification using the decision tree to find the attractor areas of the local minima (Decision Tree) are presented. The numerical comparison was performed on two classes of the GKLS functions (Simple and Hard from [16]) of dimensionalities 2, 3, 4, and 5. These two classes differ in the size of the attraction region of the global minimum point. For the simple class of problems, the radius of the attraction region is three times larger than for a complex one.

The stop criterion for the algorithms was the falling of the next trial points into the ε-nearness of true global minimum. In Tab. 2, the averaged numbers of iterations executed by the algorithms when solving the problems and the values of time speedup relative to the sequential run are compared. The parallelization was performed using MPI technology, the run was performed on 8 processes.

Table 2. Averaged numbers of iterations and averaged numbers of trials executed by different algorithms

N	Problem class	Averaged number of iterations		Speedup	
		AGSA	Decision tree	AGSA	Decision tree
2	Simple	3 823.1	485.6	7.4	12.7
	Hard	787.6	304.2	13.8	14.9
3	Simple	5 700.5	720.1	5.6	14.2
	Hard	2 411.9	634.3	5	5.9
4	Simple	31 101.5	1 037.4	5.1	7.1
	Hard	10 418.1	816.2	6.9	5
5	Simple	163 313.5	1 204.2	4.8	4.1
	Hard	27 524.5	999.4	4.6	6.3

As one can see from Tab. 2, the iteration speedup is large enough for the problems of any dimensionality.

Conclusion

So far, as a result of the work done, we succeeded to combine the global search algorithm with a local optimization method. Unlike the known multistart schemes, the decision to run the local method is made using the decision tree. The use of such a combination of methods allows considerably accelerating the algorithm.

The parallel version of the algorithm preserves the properties of its sequential prototype that was confirmed by he numerical experiments on solving a series of several hundred problems of various dimensionalities. The proposed scheme allows us utilizing the advantages of both parallelization as well as fast search of local extrema.

In addition to using decision trees to identify attraction regions for local extrema of multiextremal functions, we also plan to use machine learning methods to separate the variables of the problem being solved. In many applied optimization problems the dependence of the objective function on some parameters is either linear or unimodal. Separating such variables into a special group and solving the problem using a parallel recursive optimization scheme [17] can reduce the time for solving the problem by orders of magnitude compared to using a global search for all variables at once.

This study was supported by the Russian Science Foundation, project No. 21-11-00204.

References

1. Ferreiro A., Garcia J., Lopez-Salas J., Vazquez C. An efficient implementation of parallel simulated annealing algorithm in GPUs. J. Glob. Optim. 2013. Vol. 57, no. 3. P. 863–890. DOI: 10.1007/s10898-012-9979-z.
2. Garcia-Martinez J., Garzon E., Ortigosa P. A GPU implementation of a hybrid evolutionary algorithm: GPuEGO. J. Supercomput. 2014. Vol. 70, no. 2. P. 684–695. DOI: 10.1007/s11227-014-1136-7.
3. Langdon W. Graphics processing units and genetic programming: an overview. Soft Computing. 2011. Vol. 15, no. 8. P. 1657–1669. DOI: 10.1007/s00500-011-0695-2.
4. Evtushenko Y., Malkova V., Stanevichyus A.A. Parallel global optimization of functions of several variables. Comput. Math. Math. Phys. 2009. Vol. 49, no. 2. P. 246–260. DOI: 10.1134/S0965542509020055.
5. He J., Verstak A., Watson L., Sosonkina M. Design and implementation of a massively parallel version of DIRECT. Comput. Optim. Appl. 2008. Vol. 40, no. 2. P. 217–245. DOI: 10.1007/s10589-007-9092-2.
6. Paulavičius R., Žilinskas J., Grothey A. Parallel branch and bound for global optimization with combination of Lipschitz bounds. Optim. Method. Softw. 2011. Vol. 26, no. 3. P. 487–498. DOI: 10.1080/10556788.2010.551537.
7. Strongin R.G., Sergeyev Y.D. Global optimization with non-convex constraints. Sequential and parallel algorithms. Dordrecht: Kluwer Academic Publishers, 2000. DOI: 10.1007/978-1-4615-4677-1.
8. Hooke R., Jeeves T. "Direct Search" Solution of Numerical and Statistical Problems. J. ACM. 1961. Vol. 8, no. 2. P. 212–229. DOI: 10.1145/321062.321069.

9. Nocedal J., Wright S. Numerical Optimization. New York: Springer, 2006. DOI: 10.1007/b98874.

10. Kelley C.T. Iterative Methods for Optimization. Philadelphia: SIAM, 1999. DOI: 10.1137/1.9781611970920.

11. Barkalov K., Lebedev I. Solving multidimensional global optimization problems using graphics accelerators. Communications in Computer and Information Science. 2016. Vol. 687. P. 224–235. DOI: 10.1007/978-3-319-55669-7_18.

12. Barkalov K., Strongin R. A global optimization technique with an adaptive order of checking for constraints. Computational Mathematics and Mathematical Physics. 2002. Vol. 42, no. 9. P. 1289–1300.

13. Gergel V.P. A global optimization algorithm for multivariate functions with Lipschitzian first derivatives. Journal of Global Optimization. 1997. Vol. 10, no. 3. P. 257–281. DOI: 10.1023/A:1008290629896.

14. Himmelblau D. Applied Nonlinear Programming. New York: McGraw-Hill, 1972. 498 p.

15. Brahmbhatt S. Practical OpenCV (Technology in Action). New York: Apress, 2013. DOI: 10.1007/978-1-4302-6080-6_1.

16. Sergeyev Y., Kvasov D. Global Search Based on Efficient Diagonal Partitions and a Set of Lipschitz Constants. SIAM J. Optim. 2006. Vol. 16, no. 3. P. 910–937. DOI: 10.1137/040621132.

17. Barkalov K., Lebedev I., Kocheganova M., Gergel V. Combining local and global search in a parallel nested optimization scheme. Communications in Computer and Information Science. 2020. Vol. 1263. P. 100–112. DOI: 10.1007/978-3-030-55326-5_8.

УДК 519.853.4 DOI: 10.14529/cmse230301

ОБ ИСПОЛЬЗОВАНИИ ДЕРЕВЬЕВ РЕШЕНИЙ ДЛЯ ВЫЯВЛЕНИЯ ОБЛАСТЕЙ ПРИТЯЖЕНИЯ ЛОКАЛЬНЫХ МИНИМУМОВ В ПАРАЛЛЕЛЬНОМ АЛГОРИТМЕ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ

© 2023 К.А. Баркалов, И.Г. Лебедев, Д.И. Силенко

Нижегородский государственный университет им. Н.И. Лобачевского
(603022 Нижний Новгород, пр. Гагарина, д. 23)
E-mail: barkalov@vmk.unn.ru, ilya.lebedev@itmm.unn.ru, silenko@itmm.unn.ru

Поступила в редакцию: 31.07.2023

В работе рассматривается решение многомерных задач многоэкстремальной оптимизации с использованием деревьев решений для выявления областей притяжения локальных минимумов. Целевая функция представлена как «черный ящик», она может быть недифференцируемой, многоэкстремальной и вычислительно трудоемкой. Для функции предполагается, что она удовлетворяет условию Липшица с априори неизвестной константой. Для решения поставленной задачи многоэкстремальной оптимизации применятся алгоритм глобального поиска. Хорошо известно, что сложность решения существенно зависит от наличия нескольких локальных экстремумов. В данной работе предложена модификация алгоритма, в которой определяются окрестности локальных минимумов целевой функции на основе анализа накопленной поисковой информации. Проведение такого анализа с использованием методов машинного обучения позволяет принять решение о запуске локального метода, что может ускорить сходимость алгоритма. Данный подход был подтвержден результатами численных экспериментов, демонстрирующих ускорение при решении набора тестовых задач.

Ключевые слова: глобальная оптимизация, многоэкстремальные функции, параллельные вычисления, машинное обучение, дерево решений.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Barkalov K.A., Lebedev I.G., Silenko D.I. On Using the Decision Trees to Identify the Local Extrema in Parallel Global Optimization Algorithm // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 3. С. 5–18. DOI: 10.14529/cmse230301.

Литература

1. Ferreiro A., Garcia J., Lopez-Salas J., Vazquez C. An efficient implementation of parallel simulated annealing algorithm in GPUs // J. Glob. Optim. 2013. Vol. 57, no. 3. P. 863–890. DOI: 10.1007/s10898-012-9979-z.
2. Garcia-Martinez J., Garzon E., Ortigosa P. A GPU implementation of a hybrid evolutionary algorithm: GPuEGO // J. Supercomput. 2014. Vol. 70, no. 2. P. 684–695. DOI: 10.1007/s11227-014-1136-7.
3. Langdon W. Graphics processing units and genetic programming: an overview // Soft Computing. 2011. Vol. 15, no. 8. P. 1657–1669. DOI: 10.1007/s00500-011-0695-2.
4. Евтушенко Ю.Г., Малкова В.У., Станевичюс А.А. Параллельный поиск глобального экстремума функций многих переменных // Вычислительная математика и математическая физики. 2009. Т. 49, № 2. С. 246–260. DOI: 10.1134/S0965542509020055.

5. He J., Verstak A., Watson L., Sosonkina M. Design and implementation of a massively parallel version of DIRECT // Comput. Optim. Appl. 2008. Vol. 40, no. 2. P. 217–245. DOI: 10.1007/s10589-007-9092-2.

6. Paulavičius R., Žilinskas J., Grothey A. Parallel branch and bound for global optimization with combination of Lipschitz bounds // Optim. Method. Softw. 2011. Vol. 26, no. 3. P. 487–498. DOI: 10.1080/10556788.2010.551537.

7. Strongin R.G., Sergeyev Y.D. Global optimization with non-convex constraints. Sequential and parallel algorithms. Dordrecht: Kluwer Academic Publishers, 2000. DOI: 10.1007/978-1-4615-4677-1.

8. Hooke R., Jeeves T. "Direct Search" Solution of Numerical and Statistical Problems // J. ACM. 1961. Vol. 8, no. 2. P. 212–229. DOI: 10.1145/321062.321069.

9. Nocedal J., Wright S. Numerical Optimization. New York: Springer, 2006. DOI: 10.1007/b98874.

10. Kelley C.T. Iterative Methods for Optimization. Philadelphia: SIAM, 1999. DOI: 10.1137/1.9781611970920.

11. Barkalov K., Lebedev I. Solving multidimensional global optimization problems using graphics accelerators // Communications in Computer and Information Science. 2016. Vol. 687. P. 224–235. DOI: 10.1007/978-3-319-55669-7_18.

12. Баркалов К.А., Стронгин Р.Г. Метод глобальной оптимизации с адаптивным порядком проверки ограничений // Вычислительная математика и математическая физики. 2002. Т. 42, № 9. С. 1289–1300.

13. Gergel V.P. A global optimization algorithm for multivariate functions with Lipschitzian first derivatives // Journal of Global Optimization. 1997. Vol. 10, no. 3. P. 257–281. DOI: 10.1023/A:1008290629896.

14. Himmelblau D. Applied Nonlinear Programming. New York: McGraw-Hill, 1972. 498 p.

15. Brahmbhatt S. Practical OpenCV (Technology in Action). New York: Apress, 2013. DOI: 10.1007/978-1-4302-6080-6_1.

16. Sergeyev Y., Kvasov D. Global Search Based on Efficient Diagonal Partitions and a Set of Lipschitz Constants // SIAM J. Optim. 2006. Vol. 16, no. 3. P. 910–937. DOI: 10.1137/040621132.

17. Barkalov K., Lebedev I., Kocheganova M., Gergel V. Combining local and global search in a parallel nested optimization scheme // Communications in Computer and Information Science. 2020. Vol. 1263. P. 100–112. DOI: 10.1007/978-3-030-55326-5_8.

Баркалов Константин Александрович, д.т.н., доцент, кафедра математического обеспечения и суперкомпьютерных технологий, Нижегородский государственный университет им. Н.И. Лобачевского (Нижний Новгород, Российская Федерация)

Лебедев Илья Генадьевич, заведующий лабораторией суперкомпьютерных технологий и высокопроизводительных вычислений, Нижегородский государственный университет им. Н.И. Лобачевского (Нижний Новгород, Российская Федерация)

Силенко Дмитрий Игоревич, магистр, Нижегородский государственный университет им. Н.И. Лобачевского (Нижний Новгород, Российская Федерация)