Comparison of dimensionality reduction schemes for parallel global optimization algorithms *

Konstantin Barkalov, Vladislav Sovrasov, and Ilya Lebedev

Lobachevsky State University of Nizhny Novgorod, Nizhny Novgorod, Russia
konstantin.barkalov@itmm.unn.ru
sovrasov.vlad@gmail.com
ilya.lebedev@itmm.unn.ru
http://hpc-education.unn.ru/основные-направления/глобальная-оптимизация

Аннотация This work considers a parallel algorithms for solving multi-extremal optimization problems. Algorithms are developed within the framework of the information-statistical approach and implemented in a parallel solver "Globalizer" . The optimization problem is solved by reducing the multidimensional problem to a set of joint one-dimensional problems that are solved in parallel. Five types of Peano-type space-filling curves are employed to reduce dimension. The results of computational experiments carried out on several hundred test problems are discussed.

Keywords: Global optimization · Dimension reduction · Parallel algorithms · Multidimensional multiextremal optimization · Global search algorithms · Parallel computations

1 Introduction

Global (or multiextremal) optimization problems are among the most complex problems in both theory and practice of optimal decision making. In these kinds of problems, the optimization criterion can have several local optima within the search domain. The existence of several local optima makes finding the global optimum difficult essentially, since it requires examining the whole feasible search domain. The volume of computations for solving global optimization problems can increase exponentially with increasing number of varied parameters.

These global optimization problem features impose special requirements on the quality of the optimization methods and on the software to implement these ones. The global optimization methods should be highly efficient, and the software systems should be developed on a good professional basis. In general, the global optimization problems can be solved at a reasonable time by employing parallel computations on modern supercomputing systems only.

The general state of the art in the field of global optimization is presented in a number of key monographs [13], [25], [33], [38], [51], [52], [54]. The development of optimization methods, which use the high-performance computational systems

* The study was supported by the Russian Science Foundation, project No 16-11-10150

to solve the time-consuming global optimization problems, is an area of intensive research — see, for instance, [7], [8], [34], [50], [51]. The obtained theoretical results provide the efficient solutions of many applied global optimization problems in various fields of scientific and technical applications [10], [11], [12], [28], [29], [33], [34], [37], [38].

At the same time, the practical implementation of these global optimization algorithms within the framework of industrial software systems is quite limited. In many cases, software implementations are experimental in nature and are used by the developers themselves to obtain the results from the computational experiments required for the scientific publications. This situation originates from high development costs of the professional software systems, which can be used by numerous users. In addition, the global optimization problems could be solved in an automatic mode rarely because of the complexity of these ones. The user should actively control the global search process that implies an adequate level of qualification in the field of optimization (particularly, the user should know and understand the global optimization methods well).

In this work, the authors consider an approach to minimizing multiextremal functions developed in ...???????? This allows problems to be solved in which function values may not be determined for the entire search domain. Under this approach, solving multidimen-sional problems is reduced (using Peano-type space-filling curves) to solving equivalent one-dimensional problems. It should be noted that standard approaches to algorithm parallelization are not quite applicable to global optimization. For example, the rules for selecting an-other iteration point are quite simple and do not require parallelization (as overheads associated with organizing parallel computations will nullify any possible acceleration). Some acceleration can be achieved by parallelizing the computation of function values describing the object to be optimized; however, this ap-proach is specific to each individual problem being solved. The following approach looks more promising. The algorithm can be modified to run several trials in parallel. This approach provides the efficiency (as paral-lelization is applied to the most computation-intensive part of the problem solv-ing process) and generality (in that it applies to a wide range of global optimiza-tion algorithms). The approach, described in [55] for unconstrained optimization, was used in this work for parallelizing constrained optimization algorithms.

2 Statement of Multidimensional Global Optimization Problem

In this paper, the core class of optimization problems, which can be solved using Globalizer[61], is formulated. This class involves the multidimensional global optimization problems without constraints, which can be defined in the following way:

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

with the given boundary vectors a and b. It is supposed, that the objective function $\varphi(y)$ satisfies the Lipschitz condition

$$|\varphi(y_1) - \varphi(y_2)| \leq L\|y_1 - y_2\|, y_1, y_2 \in D, \tag{2}$$

where $L > 0$ is the Lipschitz constant, and $||\cdot||$ denotes the norm in $\mathbf{R}^N$ space.

Usually, the minimized function $\varphi(y)$ is defined as a computational procedure, according to which the value $\varphi(y)$ can be calculated for any vector $y \in D$ (let us further call such a calculation *a trial*). It is supposed that this procedure is a time-consuming one. As a result, the overall time of solving the optimization problem (1) is determined, first of all by the number of executed trials. It should also be noted that the requirement of the Lipschitz condition (2) is highly important, since an estimate of the global minimum can be constructed on the basis of a finite number of computed values of the optimized function only in this case .

As it has been shown earlier by many researchers (see, for instance, [12], [25], [38], [51]), finding the numerical estimate of the global optimum implies constructing a coverage of the search domain D. As a result, the computational costs of solving the global optimization problems are readily very high even for a small number of varied parameters (the dimensionality of the problem). A notable reduction in the volume of computations can be achieved when the coverage of the search domain is non-uniform, i. e. the series of trial points is only dense in a vicinity of the global optimum point. The construction of such a non-uniform coverage could be provided in an adaptive way, when the selection of the next trial points is determined by using the search information (the preceding trial points and the values of the minimized function at these points) obtained in the course of computations. This necessary condition complicates considerably the computational schemes of global optimization methods since it implies a complex analysis of a large amount of multidimensional search information. As a result, many optimization algorithms use various approaches to the dimensional reduction [38], [46], [48], [50], [51].

3 Methods of Dimension Reduction

3.1 Базовый алгоритм

Within the framework of the information-statistical global optimization theory, the Peano space-filling curves (or evolvents) $y(x)$ mapping the interval $[0, 1]$ onto an N-dimensional hypercube D unambiguously are used for the dimensionality reduction [46], [48], [50], [51].

As a result of the reduction, the initial multidimensional global optimization problem (1) is reduced to the following one-dimensional problem:

$$\varphi(y(x^*)) = \min\{\varphi(y(x)) : x \in [0, 1]\}. \tag{3}$$

It is important to note that this dimensionality reduction scheme transforms the minimized Lipschitzian function from (1) to the corresponding one-dimensional

function $\varphi(y(x))$, which satisfies the uniform Hölder condition, i. e.

$$|\varphi(y(x_1)) - \varphi(y(x_2))| \leq H|x_1 - x_2|^{\frac{1}{N}}, x_1, x_2 \in [0,1], \tag{4}$$

where the constant H is defined by the relation $H = 2L\sqrt{N+3}$, L is the Lipschitz constant from (2), and N is the dimensionality of the optimization problem (1).

The algorithms for the numerical construction of the Peano curve approximations are given in [51]. As an illustration, an approximation of the Peano curve for the third density level is shown in Figure 1. Figure 1 demonstrates the movement order in a two-dimensional domain to construct the Peano curve approximation; the precision of the Peano curve approximation is determined by the density level used in the construction.

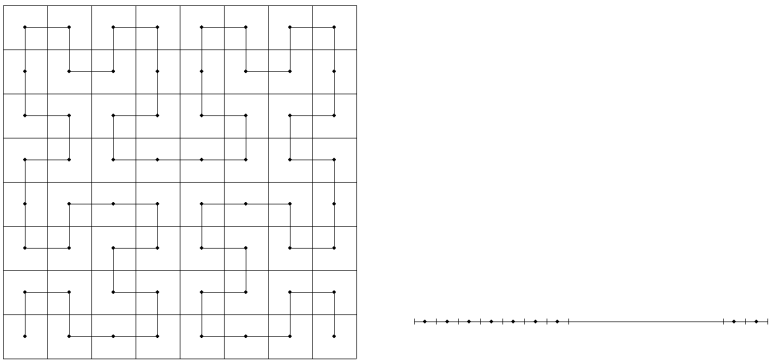


Рис. 1: A Peano curve approximation for the third density level

The computational scheme obtained as a result of the dimensionality reduction consists of the following:

- The optimization algorithm performs the minimization of the reduced one-dimensional function $\varphi(y(x))$ from (3),
- After determining the next trial point x, a multidimensional image y is calculated by using the mapping $y(x)$,
- The value of the initial multidimensional function $\varphi(y)$ is calculated at the point $y \in D$,
- The calculated value $z = \varphi(y)$ is used further as the value of the reduced one-dimensional function $\varphi(y(x))$ at the point x.

3.2 Сдвиговые

The reduction of the multidimensional problems to the one-dimensional ones using the Peano curves has such important properties as the continuity and

the uniform boundedness of the function differences for limited variation of argument. However, a partial loss of information on the nearness of the points in the multidimensional space takes place since a point $x \in [0,1]$ has the left and the right neighbors only while the corresponding point $y(x) \in D \subset R^N$ has the neighbors in $2N$ directions. As a result, when using the mappings like Peano curve the images y', y'', which are close to each other in the N-dimensional space can correspond to the preimages x', x'', which can be far away from each other in the interval $[0,1]$. This property results in the excess computations since several limit points x', x'' of the trial sequence generated by the index method in the interval $[0,1]$ can correspond to a single limit point y in the N-dimensional space.

One of the possible ways to overcome this disadvantage consists in using the multiple mappings

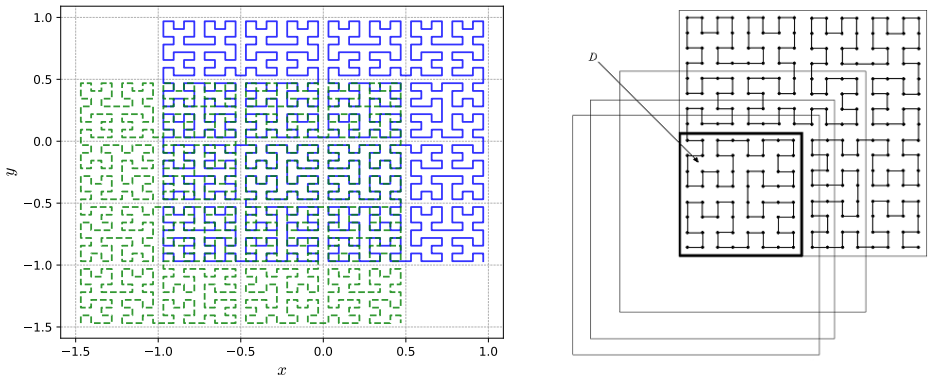
$$Y_L(x) = \left\{y^0(x),\ y^1(x), ...,\ y^L(x)\right\} \tag{5}$$

instead of single Peano curve $y(x)$ (see [56,57,51]).

Let us consider a family of the hypercubes

$$D_l = \left\{y \in R^N : -2^{-1} \le y_i + 2^{-l} \le 3 \cdot 2^{-1},\ 1 \le i \le N\right\},\ 0 \le l \le L, \tag{6}$$

where the hypercube D_{l+1} is obtained by the translation of the hypercube D_l along the main diagonal with displacement equal to 2^{-l} in each coordinate. In Fig. 2a the hypercubes $D_0, ..., D_3$ for the case $N = 2,\ L = 3$ are presented.



(a) Two mappings (b) Hypercubes D_l

Рис. 2: Multiple mappings

Let us assume that the evolvent $y^0(x)$ maps the interval $[0,1]$ onto the hypercube D_0 from (6), i.e.,

$$D_0 = \left\{y^0(x) : x \in [0,1]\right\}.$$

Then, the evolvents $y^l(x) = \{y_1^l(x), ..., y_N^l(x)\}$, the coordinates of which are defined by the conditions

$$y_i^l(x) = y_i^{l-1}(x) + 2^{-l},\ 1 \le i \le N,\ 1 \le l \le L,$$

map the interval $[0,1]$ onto the corresponding hypercubes $D_l,\ 1 \le l \le L$. In Fig. 2a the image of the interval $[0,1]$ obtained by the curve $y^0(x),\ x \in [0,1]$, is shown as the dashed line. Since the hypercube D from (1) is included in the common part of the family of hypercubes (6) (the boundaries of hypercube D are highlighted in Fig. 2b), having introduced an additional constraint function

$$g_0(y) = \max\left\{|y_i| - 2^{-1} :\ 1 \le i \le N\right\}, \tag{7}$$

one can present the initial hypercube D in the form

$$D = \left\{y^l(x) :\ x \in [0,1],\ g_0(y^l(x)) \le 0\right\},\ 0 \le l \le L,$$

i.e., $g_0(y) \le 0$ if $y \in D$ and $g_0(y) > 0$ otherwise. Consequently, any point $y \in D$ has its own preimage $x^l \in [0,1]$ for each mapping $y^l(x),\ 0 \le l \le L$.

Thus, each evolvent $y^l(x),\ 0 \le l \le L$, generates its own problem of the type (1) featured by its own extended (in comparison with D) search domain D_l and the additional constraint with the left hand part from (7)

$$\min\left\{\varphi(y^l(x)) : x \in [0,1],\ g_j(y^l(x)) \le 0,\ 0 \le j \le m\right\},\ 0 \le l \le L. \tag{8}$$

The problems (8) correspond to the domains $Q_0^l = [0,1]$ and $Q_{j+1}^l, 0 \le j \le m$, defined by the expression

$$Q_{j+1}^l = \left\{x \in Q_j^l : g_j(y^l(x)) \le 0\right\},\ 0 \le j \le m,$$

for the corresponding evolvents $y^l(x)$. The application of a multiple mapping defines the following relation for the nearness in the multidimensional search domain and in the one-dimensional one.

Theorem 1. *Let a point y^* from the domain D be contained in the line segment with the end-points $y', y'' \in D$ meeting the requirements*

$$\left|y_j' - y_j''\right| \le 2^{-p},\ y_i' = y_i'' = y_i^*,\ 1 \le i \le N,\ i \ne j,$$

where p is an integer and $1 \le p \le L$, i.e., the line segment is collinear with the j-th axis in R^N. Then, there exist at least one mapping $y^l(x),\ 0 \le l \le L$, and the preimages $x^*,\ x',\ x'' \in [0,1]$ such that

$$y^* = y^l(x^*),\ y' = y^l(x'),\ y'' = y^l(x'')$$

and

$$\max\left\{|x' - x^*|,\ |x'' - x^*|,\ |x' - x''|\right\} \le 2^{-pN}.$$

Доказательство. Proof of this theorem is given in [51]. □

The conditions of the theorem single out a specific vicinity of the point y^*. This vicinity includes only the points, which can be obtained by the shift of y^* parallel to one of the coordinate axes with a displacement not more than 2^{-p}. By changing j, $1 \leq j \leq N$, in the theorem conditions it is possible to obtain the neighbours in any N coordinate directions. According to the statement, the closeness of the points in the N-dimensional space in a particular direction will be reflected by the closeness of their preimages in one of the univariate problems. The information on the closeness of the points results, first, in more precise estimate of Lipschitz constants and, second, in the increase of the characteristics of the intervals, the images of the end points of which are close to each other in the N-dimensional space.

3.3 Вращаемые

The application of the scheme for building the multiple evolvents (hereinafter called the shifted evolvents or S-evolvents) described in Subsection 3.2 allows to preserve the information on the nearness of the points in the multidimensional space and, therefore, to provide more precise (as compared to a single evolvent) estimate of Lipschitz constant in the search process. However, this approach has serious restrictions, which narrow the applicability of the parallel algorithms, designed on the base of the S-evolvents.

Because a shifted evolvent is built as a result of the shift of a hypercube D along the main diagonal, and the shift step decreases 2 times for the building of each subsequent mapping, the number of such shifts is limited by the evolvent density m. In the case when the shift step is less than 2^{-m}, the next evolvent coincides with the previous one. Thus, the applicable number of the evolvents L and, hence, the number of processors are limited by m ($L \leq m$), where m is the density of the evolvents.

Also, it follows from the algorithm for building the shifted evolvents that solving the initial problem in the domain D is reduced to solving a set of problems in the extended domains D_l from (6) with additional constraint (7). As a result, the structure of the search domain becomes more complicated, and the exponential decrease of the volume of the search domain D in relation to the volumes of the domains D_l with increasing dimensionality of the problem takes place.

To overcome the disadvantages mentioned above and to preserve the information on the nearness of the points in the N-dimensional space, a novel scheme of building of the multiple mappings is proposed. The building of a set of Peano curves not by the shift along the main diagonal of the hypercube but by rotation of the evolvents around the coordinate origin is a distinctive feature of the proposed scheme [58]. In the initial non-rotated mapping for close points y', y'' in the multidimensional space their preimages x', x'' in the interval $[0, 1]$ can be far away from each other. In the rotated scheme there exists a mapping $y^i(x)$ according to which preimages x', x'' will be located nearer. The evolvents generated according to the novel scheme will be hereinafter called the rotated evolvents or R-evolvents.

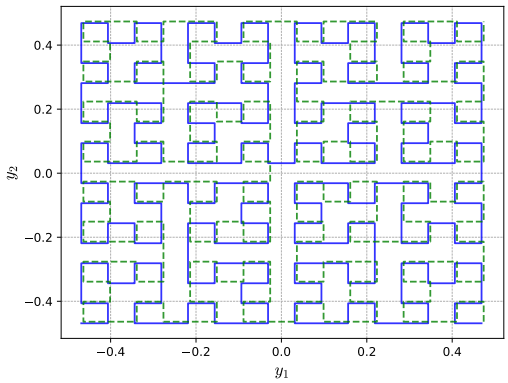


Рис. 3: Two rotated evolvents on the same plane

In Fig. 3 two evolvents being the approximations to Peano curves for the case $N = 2$ are presented as an illustration.

Taking into account the initial mapping, one can conclude that current implementation of the method allows to build up to $N(N-1)+1$ evolvents for mapping the N-dimensional domain onto the corresponding one-dimensional intervals. Moreover, the additional constraint $g_0(y) \leq 0$ with $g_0(y)$ from (7), which arises in shifted evolvents, is absent. This method for building a set of mappings can be "scaled" easily to obtain more evolvents (up to 2^N) if necessary.

The use of the set of mappings $Y_L(x) = \{y^1(x), ..., y^L(x)\}$ results in the appearing of the corresponding set of the one-dimensional multiextremal problems

$$\min\left\{\varphi(y^l(x)) : x \in [0,1],\ g_j(y^l(x)) \leq 0,\ 1 \leq j \leq m\right\},\ 1 \leq l \leq L. \qquad (9)$$

Each problem from this set can be solved independently. Any computation of the value $z = g_\nu(y')$, $y' = y^i(x')$ of the function $g_\nu(y)$ in the i-th problem can be interpreted as a computation of the value $z = g_\nu(y')$, $y' = y^s(x'')$ for any other s-th problem without time-comsuming computations of the functions $g_\nu(y)$. Such information integrity allows to solve initial problem (1) by solving L problems (9) on a set of intervals $[0,1]$ in parallel by the index method.

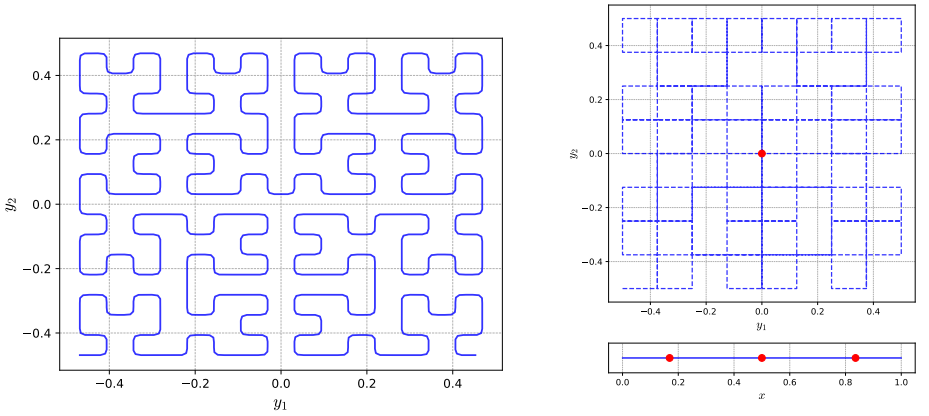
3.4 Non injective evolvent

Как уже было сказано в секции 3.2, потеря информации о близости точек в многомерном пространстве может быть частично скомпенсирована использованием множественных отображений $Y_L(x) = \{y^1(x), ..., y^L(x)\}$. Однако, сама по себе кривая типа Пеано сохраняет в себе часть этой информации: она не является инъективым отображением, поэтому имея один образ $y(x) \in \mathbb{R}^N$, можно получить несколько несколько отличных x прообразов $t_j \in [0,1], t_j \neq x$, которые затем могут быть добавлены в поисковую информацию индексного метода.

Кривая типа Пеано, используемая в (3) для редукции размерности, определяется через предельный переход, поэтому не может быть вычислена непосредтвенно. При численной оптимизации используется некоторое её приближение, являющееся инъективной кусочно-линейной кривой. В [48] было предложено неинъективное отображение равномерной сетки на отрезка $[0,1]$ на равномерную сетку в гиперкубе D. Каждый многомерный узел может иметь до 2^N одномерных прообразов. На рис. 4b изображена кривая с самопересечениями, полученная при соединении узлов грубой многомерной сетки в порядке следования их прообразов из отрезка $[0,1]$, а также отмечана точка, имеющая 3 прообраза.

Недостатком неинъективной развёртки является потенциально большое количетво прообразов (до 2^N) и невозможность использования параллельной схемы для множественных отображений из секции 4.2.

3.5 Smooth evolvent



(a) Smooth evolvent

(b) Numerical non injective evolvent

Рис. 4: Different evolvents

4 Parallel Computations for Solving Global Optimization Problems.

4.1 Core Multidimensional Algorithm of Global Search

The information-statistical theory of global optimization formulated in [48], [51] has served as a basis for the development of a large number of efficient

multiextremal optimization methods [1], [18], [19], [23], [30], [42], [43], [45], [46], [47].

The optimization methods applied in Globalizer are based on the MAGS method, which can be presented as follows — see [48], [51].

Let us introduce a simpler notation for the optimization problem being solved

$$f(x) = \varphi(y(x)) : x \in [0,1]. \tag{10}$$

The initial iteration of the algorithm is performed at an arbitrary point $x^1 \in (0,1)$. Then, let us suppose that k, $k \geq 1$, optimization iterations have been completed already. The selection of the trial point x^{k+1} for the next iteration is performed according to the following rules.

Rule 1. Renumber the points of the preceding trials by the lower indices in order of increasing value of coordinates

$$0 = x_0 < x_1 < ... < x_{k+1} = 1, \tag{11}$$

the points x_0, x_{k+1} were introduced additionally for the convenience of further explanation, the values of the minimized function z_0, z_{k+1} at these points are undefined.

Rule 2. Compute the current estimate of the Hölder constant H from (4)

$$M = \max_{1<i\leq k} \frac{|z_i - z_{i-1}|}{\rho_i},\ m = \begin{cases} rM, & M > 0, \\ 1, & M = 0, \end{cases} \tag{12}$$

as the maximum of the relative differences of the minimized function values on the set of previously executed trial points $x_i, 1 \leq i \leq k$ from (11). Hereafter, $\rho_i = (x_i - x_{i-1})^{\frac{1}{N}}, 1 \leq i \leq k+1$. The constant r, $r > 1$, is the reliability parameter of the algorithm.

Rule 3. Compute the characteristics $R(i)$ for each interval $(x_{i-1}, x_i), 1 \leq i \leq k+1$, where

$$\begin{gathered} R(i) = 2\rho_i - 4\frac{z_i}{m}, \quad i = 1, \\ R(i) = \rho_i + \frac{(z_i - z_{i-1})^2}{m^2\rho_i} - 2\frac{z_i + z_{i-1}}{m}, \quad 1 < i < k+1, \\ R(i) = 2\rho_i - 4\frac{z_{i-1}}{m}, \quad i = k+1. \end{gathered} \tag{13}$$

Rule 4. Determine the interval with the maximum characteristic

$$R(t) = \max_{1\leq i\leq k+1} R(i). \tag{14}$$

Rule 5. Execute a new trial at the point x^{k+1} located within the interval with the maximum characteristic from (14)

$$x^{k+1} = \frac{x_t + x_{t-1}}{2} - \mathrm{sign}(z_t - z_{t-1})\frac{1}{2r}\left[\frac{r|z_t - z_{t-1}|}{m}\right]^N, \ \text{ if } 1 < t < k+1, \tag{15}$$

$$x^{k+1} = \frac{x_t + x_{t-1}}{2}, \text{ if } t = 1 \text{ or } t = k+1.$$

The stopping condition, which terminated the trials, is defined by the inequality

$$\rho_t < \varepsilon \tag{16}$$

for the interval with the maximum characteristic from (14) and $\varepsilon > 0$ is the predefined accuracy of the optimization problem solution. If the stopping condition is not satisfied, the index k is incremented by 1, and the new global optimization iteration is executed.

In order to explain the algorithm presented above, let us note the following. The characteristics $R(i), 1 \leq i \leq k+1$, calculated according to (13) could be interpreted as some measures of importance of the intervals with respect to the expected location of the global minimum point. Thus, the rules (14) and (15) for selecting the interval of the next trial become more clear — the point of every next global optimization iteration is selected within the interval, where the global minimum point can be found most likely.

The convergence conditions of the described algorithm are given, for example, in [51].

4.2 Параллельные множественные отображения

Using the multiple mapping allows solving initial problem (1) by parallel solving the problems

$$\min\{\varphi(y^s(x)) : x \in [0,1]\}, 1 \leq s \leq S$$

on a set of intervals [0,1] by the index method. Each one-dimensional problem is solved on a separate processor. The trial results at the point x^k obtained for the problem being solved by particular processor are interpreted as the results of the trials in the rest problems (in the corresponding points $x^(k_1), \ldots, x^(k_S)$). In this approach, a trial at the point $x^k \in [0,1]$ executed in the framework of the s-th problem, consists in the following sequence of operations.

1. Determine the image $y^k = y^s(x^k)$ for the evolvent $y^s(x)$.

2. Inform the rest of processors about the start of the trial execution at the pointy^k (the blocking of the point y^k).

3. Determine the preimages $x^{k_s} \in [0,1], 1 \leq s \leq S$, of the point y^k and interpret the trial executed at the point $y^k \in D$ as the execution of the trials in the S points $x^{k_1}, \ldots, x^{k_s}$

4. Inform the rest of processors about the trial results at the point y^k.

The decision rules for the proposed parallel algorithm, in general, are the same as the rules of the sequential algorithm (except the method of the trial execution). Each processor has its own copy of the software realizing the computations of the problem functions and the decision rule of the index algorithm. For the organization of the interactions among the processors, the queues are created on each processor, where the processors store the information on the executed iterations in the form of the tuples: the processor number s, the trial point x^{k_s}.

The proposed parallelization scheme was implemented with the use of MPI technology. Main features of implementation consist in the following. A separate MPI-process is created for each of S one-dimensional problems being solved, usually, one process per one processor employed. Each process can use p threads, usually one thread per an accessible core.

At every iteration of the method, the process with the index $s, 0 \leqslant s < S$ performs p trials in parallel at the points $x^{(s+i_S)}, 0 \leqslant i < p$. At that, each process stores all S_p points, and an attribute indicating whether this point is blocked by another process or not is stored for each point. Let us remind that the point is blocked if the process starts the execution of a trial at this point.

At every iteration of the algorithm, operating within the s-th process, determines the coordinates of p «its own» trial points. Then, the interchange of the coordinates of images of the trial points $y^{(s+i_S)}, 0 \leqslant i < p, 0 \leqslant s < S$ is performed (from each process to each one). After that, the preimages $x^{(q+i_S)}, 0 \leqslant q < S, q \neq s$ of the points received by the s-th process from the neighbor ones are determined with the use of the evolvent $y^s(x)$. The points blocked within the s-th process will correspond to the preimages obtained. Then, each process performs the trials at the non-blocked points, the computations are performed in parallel using OpenMP. The results of the executed trials (the index of the point, the computed values of the problem functions, and the attribute of unblocking of this point) are transferred to all rest processes. All the points are added to the search information database, and the transition to the next iteration is performed.

5 Results of Numerical Experiments

The computational experiments have been carried out on the Lobachevsky supercomputer at State University of Nizhny Novgorod. A computational node included 2 Intel Sandy Bridge E5-2660 2.2 GHz processors, 64 GB RAM. The CPUs had 8 cores (i. e. total 16 cores were available per a node). Все рассматриваемые алгоритмы и развёртки были реализованы на языке C++ в рамках программной системы Globalizer[61]. Для реализации параллелизма на одном узле использована технология OpenMP, а для параллелизма на несколько узлов — стандарт MPI.

Сравнение алгоритмов глобальной оптимизации проведено путём оценки качества решения алгоритмами выборки задач из некоторого тестового класса. В данной статье рассматривается тестовый класс, порождаемый генератором GKLS [59]. Данные генератор позволяет конструировать сложные многоэкстремальные задачи различной размерности. В работе рассматриваются выборки по 100 задач классов размерности 2, 3, 4, 5. Каждый класс имеет две степени сложности — *Simple* и *Hard*. Параметры генератора для рассматриваемых классов приведены в [59].

In order to evaluate the efficiency of an algorithm on a given set of 100 problems, we will use the operating characteristics [60], which are defined by a set of points on the (K, P) plane where K is the average number of search trials conducted before satisfying the termination condition when minimizing

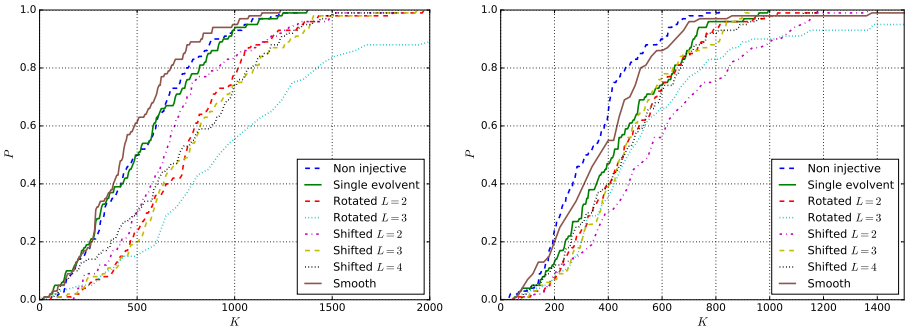
a function from a given class, and P is the proportion of problems solved successfully. If at a given K, the operating characteristic of a method goes higher than one from another method, it means that at fixed search costs, the former method has a greater probability of finding the solution. If some value of P is fixed, and the characteristic of a method goes to the left from that of another method, the former method requires fewer resources to achieve the same reliability.

5.1 Сравнение последовательных разверток

С целью понять, обладает ли какой-либо из перечисленных ранее типов развёрток существенным преимуществом над другими, были построены операционные характеристики индексного метода с различными типами развёрток на классах GKLS 2d Simple и GKLS 3d Simple. The global minimum was considered to be found if the algorithm generates a trial point y^k in the δ-vicinity of the global minimizer, i.e. $\left\|y^k - y^*\right\|_\infty \leq \delta$. The size of the vicinity was selected as $\delta = 0.01 \left\|b - a\right\|_\infty$. In case of GKLS $\delta = 0.01$.

Во всех экспериментах параметр плотности построения развёрток $m = 12$. Минимальное значение пераметра надёжности r было найдено для каждого типа развёртки перебором по равномерной сетке с шагом 0.1.

На классе GKLS 2d Simple при минимальном r неинъективная и гладкая развёртка обеспечивают более быструю сходимость (рис. 5b). То же самое, наблюдается и при $r = 5.0$ (рис. 5a). В последнем случае сдвиговая и вращаемая развёртки начинают отставать от остальных, т.к. значение $r = 5.0$ является завышенным для них.

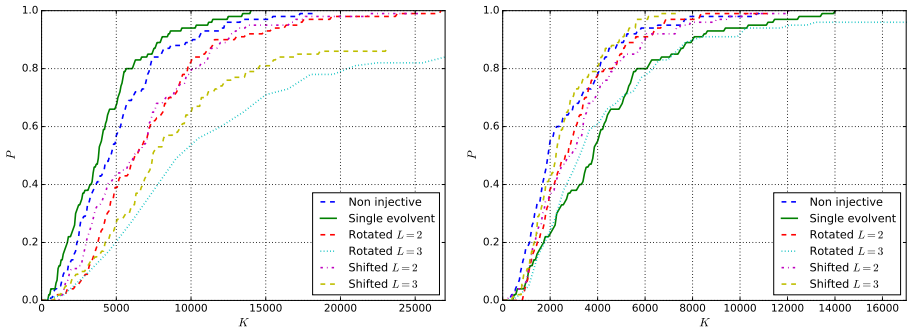


(a) $r = 5.0$ (b) Minimal r

Рис. 5: Operating characteristics on GKLS 2d Simple class

На классе GKLS 2d Simple при минимальном r неинъективная и множественные развёртки имеют значительное преимущество над единственной

развёрткой (рис. 6b). Значение $r = 4.5$ является завышенным для вращаемой и сдвиговой развёрток (рис. 6a).



(a) $r = 4.5$ (b) Minimal r

Рис. 6: Operating characteristics on GKLS 3d Simple class

Накладные расходы при использовании сдвиговой развёртки. Во всех представленных выше экспериментах при построении операционной характеристики учитывалось количество вычислений целевой функции из класса GKLS, однако в случае сдвиговой развёртки индексный метод решает задачу с ограничением g_0 из (7). В точках, где g_0 нарушено, значение целевой функции не вычисляется. Эти точки, тем не менее, хранятся в поисковой информации, создавая дополнительные расходы вычислительных ресурсов. В таблице 1 приведено среднее количество обращение к g_0 и целевой функции. При $L = 3$ ограниение g_0 вычисляется почти в 20 раз чаще, чем целевая функция φ, т. е. 95% всей поисковой информации приходится на вспомогательные точки. Такие накладные расходы приемлемы при решении задач малой размерности с трудоёмкими целевыми функциями, но при росте размерности и общего количества испытаний выгоднее использовать другие типы развёрток.

Таблица 1: Среднее количество вычислений g_0 и φ при решении задач класса GKLS 3d Simple с помощью сдвиговой развёртки

L	$calc(g_0)$	$calc(\varphi)$	$\frac{calc(g_0)}{calc(\varphi)}$ ratio
2	96247.9	6840.14	14.07
3	153131.0	7702.82	19.88

5.2 Параллельные вращаемые развертки

Чтобы оценить эффективность параллельного алгортима из секции 4.2 были проведены вычислительные эксперименты на классах GKLS 4d (Hard, Simple) и GKLS 5d (Hard, Simple). Значения r во всех экспериментах равно 5.0, размер δ-окрестности известного решения увеличен до 0.3. При решении серий задач использовалось до 8 узлов кластера и до 32 вычислительных потоков на каждом узле.

В таблице 2 приведено среднее количество итераций при решении 100 задач из каждого из рассматриваемых классов. При увеличении числа узлов и числа потоков на каждом узле количество итераций заметно сокращается (за исключением класса GKLS 4d Simple при переходе с 1 узла на 4 узла в однопоточном режиме).

Таблица 2: Averaged numbers of iterations executed by the parallel algorithm for solving the test optimization problems

		p	$N=4$		$N=5$	
			Simple	*Hard*	*Simple*	*Hard*
I	**1 cluster node**	*1*	12167	25635	20979	187353
		32	328	1268	898	12208
II	**4 cluster nodes**	*1*	25312	11103	1472	17009
		32	64	913	47	345
III	**8 cluster nodes**	*1*	810	4351	868	5697
		32	34	112	35	868

Если считать, что затраты на параллелизм пренебрежимо малы по сравнению с затратами на вычисление целевых функций в задачах оптимизации, то ускорение по времени от использования параллельного метода будет равно ускорению по итерациям. Однако, в действительности это предположение не всегда справедливо. Во всех численных экспериментах время вычисления целевой функции занимает примерно 10^{-3}с. В таблице 3 приведено ускорение по итерациям и в куглых скобках ускорение по времени. В первой строке таблицы, соответствующей последовательному режиму, в скобках приведено среднее время решения одной задачи. Из таблицы видно, что для класов GKLS 4d выгоднее использовать один узел в многопоточном режиме, тогда как для решения более сложных пятимерных задач лучше использовать несколько узлов, каждый из которых работает в параллельном режиме.

Таблица 3: Speedup of parallel computations executed by the parallel algorithm

		p	$N=4$		$N=5$	
			Simple	*Hard*	*Simple*	*Hard*
I	**1 cluster node**	*1*	12167(10.58s)	25635(22.26s)	20979(22.78s)	187353(205.83s)
		32	37.1(18.03)	20.2(8.55)	23.3(8.77)	15.4(9.68)
II	**4 cluster nodes**	*1*	0.5(0.33)	2.3(0.86)	14.3(6.61)	11.0(6.06)
		32	190.1(9.59)	28.1(1.08)	446.4(19.79)	543.0(43.60)
III	**8 cluster nodes**	*1*	15.0(6.05)	5.9(2.36)	24.2(17.56)	32.9(24.87)
		32	357.9(2.36)	228.9(2.64)	582.8(20.96)	793.0(33.89)

6 Вывод о целесообразности применения того или иного вида разверток для того или иного вида задач

Список литературы

1. K. Barkalov and V. Gergel, Multilevel scheme of dimensionality reduction for parallel global search algorithms, in *Proceedings of the 1st International Conference on Engineering and Applied Sciences Optimization*, (2014), 2111–2124.
2. K. Barkalov and V. Gergel, Parallel global optimization on GPU, *J. Glob. Optim.*, **66** (2016), 3–20.
3. K. Barkalov, V. Gergel and I. Lebedev, Use of Xeon Phi coprocessor for solving global optimization problems, *LNCS*, **9251** (2015), 307–318.
4. K. Barkalov, V. Gergel, I. Lebedev and A. Sysoev, Solving the global optimization problems on heterogeneous cluster systems, in *CEUR Workshop Proceedings*, **1482** (2015), 411–419.
5. K. Barkalov, A. Polovinkin, I. Meyerov, S. Sidorov and N. Zolotykh, SVM regression parameters optimization using parallel global search algorithm, *LNCS*, **7979** (2013), 154–166.
6. (MR3618583)M. R. Bussieck and A. Meeraus, General algebraic modeling system (GAMS), in *Modeling Languages in Mathematical Optimization* (ed. J. Kallrath), Springer, (2004), 137–157.
7. Y. Censor and S. A. Zenios, *Parallel Optimization: Theory, Algorithms, and Applications*, Oxford University Press, 1998.
8. (MR2499546)R. Čiegis, D. Henty, B. Kågström and J. Žilinskas, *Parallel Scientific Computing and Optimization: Advances and Applications*, Springer, 2009.
9. I. N. Egorov, G. V. Kretinin, I. A. Leshchenko and S. V. Kuptzov, IOSO optimization toolkit — novel software to create better design, in *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2002. Available from http://www.iosotech.com/text/2002_4329.pdf.
10. D. Famularo, P. Pugliese and Y. D. Sergeyev, A global optimization technique for checking parametric robustness, *Automatica*, **35** (1999), 1605–1611.
11. G. Fasano and J. D. Pintér, *Modeling and Optimization in Space Engineering*, Springer, 2013.

12. C. A. Floudas and M. P. Pardalos, *State of the Art in Global Optimization: Computational Methods and Applications*, Kluwer Academic Publishers, Dordrecht, 1996.
13. C. A. Floudas and M. P. Pardalos, *Recent Advances in Global Optimization*, Princeton University Press, 2016.
14. J. M. Gablonsky and C. T. Kelley, A locally-biased form of the DIRECT algorithm, *J. Glob. Optim.*, **21** (2001), 27–37.
15. M. Gaviano, D. E. Kvasov, D. Lera and Y. D. Sergeev, Software for generation of classes of test functions with known local and global minima for global optimization, *ACM Trans. Math. Software*, **29** (2003), 469–480.
16. V. Gergel and I. Lebedev, Heterogeneous parallel computations for solving global optimization problems, *Procedia Comput. Sci.*, **66** (2015), 53–62.
17. V. Gergel, A software system for multi-extremal optimization, *Eur. J. Oper. Res.*, **65** (1993), 305–313.
18. V. Gergel, A method for using derivatives in the minimization of multiextremum functions, *Comput. Math. Math. Phys.*, **36** (1996), 729–742.
19. V. Gergel, A global optimization algorithm for multivariate functions with Lipschitzian first derivatives, *J. Glob. Optim.*, **10** (1997), 257–281.
20. V. Gergel, et al., High performance computing in biomedical applications, *Procedia Computer Science*, **18** (2013), 10–19.
21. V. Gergel, et al., Recognition of surface defects of cold-rolling sheets based on method of localities, *International Review of Automatic Control,* **8** (2015), 51–55.
22. V. Gergel and S. Sidorov, A two-level parallel global search algorithm for solving computationally intensive multi-extremal optimization problems, *LNCS*, **9251** (2015), 505–515.
23. V. A. Grishagin and R. G. Strongin, Optimization of multi-extremal functions subject to monotonically unimodal constraints, *Engineering Cybernetics*, **5** (1984), 117–122.
24. K. Holmstrm and M. M. Edvall, The TOMLAB optimization environment, *Modeling Languages in Mathematical Optimization*, Springer, (2004), 369–376.
25. R. Horst and H. Tuy, *Global Optimization: Deterministic Approaches*, Springer-Verlag, Berlin, 1990.
26. (MR1246501) D. R. Jones, C. D. Perttunen and B. E. Stuckman, Lipschitzian optimization without the Lipschitz constant, *J. Optim. Theory Appl.*, **79** (1993), 157–181.
27. R. B. Kearfott, GlobSol user guide, *Optim. Methods Softw.*, **24** (2009), 687–708.
28. D. E. Kvasov and Y. D. Sergeyev, Deterministic approaches for solving practical black-box global optimization problems, *Adv. Eng. Softw.*, **80** (2015), 58–66.
29. D. E. Kvasov, D. Menniti, A. Pinnarelli, Y. D. Sergeyev and N. Sorrentino, Tuning fuzzy power-system stabilizers in multi-machine systems by global optimization algorithms based on efficient domain partitions, *Electric Power Systems Research*, **78** (2008), 1217–1229.
30. D. E. Kvasov, C. Pizzuti and Y. D. Sergeyev, Local tuning and partition strategies for diagonal GO methods, *Numerische Mathematik*, **94** (2003), 93–106.
31. L. Liberti, Writing global optimization software, in *Nonconvex Optimization and Its Applications*, Springer, **84** (2006), 211–262.
32. Y. Lin and L. Schrage, The global solver in the LINDO API, *Optim. Methods Softw.*, **24** (2009), 657–668.
33. M. Locatelli and F. Schoen, *Global Optimization: Theory, Algorithms and Applications*, SIAM, 2013.

34. G. Luque and E. Alba, *Parallel Genetic Algorithms. Theory and Real World Applications*, Springer-Verlag, Berlin, 2011.
35. M. Mongeau, H. Karsenty, V. Rouzé and J. B. Hiriart-Urruty, Comparison of public-domain software for black box global optimization, *Optim. Methods Softw.*, **13** (2000), 203–226.
36. K. M. Mullen, Continuous global optimization in R, *J. Stat. Softw.*, **60** (2014).
37. M. P. Pardalos, A. A. Zhigljavsky and J. Žilinskas, *Advances in Stochastic and Deterministic Global Optimization*, Springer, 2016.
38. J. D. Pintér, *Global Optimization in Action (Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications)*, Kluwer Academic Publishers, Dordrecht, 1996.
39. J. D. Pintér, Software development for global optimization, *Lectures on Global Optimization. Fields Institute Communications*, **55** (2009), 183–204.
40. L. M. Rios and N. V. Sahinidis, Derivative-free optimization: a review of algorithms and comparison of software implementations, *J. Glob. Optim.*, **56** (2013), 1247–1293.
41. N. V. Sahinidis, BARON: A general purpose global optimization software package, *J. Glob. Optim.*, **8** (1996), 201–205.
42. Y. D. Sergeyev, An information global optimization algorithm with local tuning, *SIAM J. Optim.*, **5** (1995), 858–870.
43. Y. D. Sergeyev, Multidimensional global optimization using the first derivatives, *Comput. Math. Math. Phys.*, **39** (1999), 743–752.
44. Y. D. Sergeyev and D. E. Kvasov, Global search based on efficient diagonal partitions and a set of Lipschitz constants, *SIAM Journal on Optimization*, **16** (2006), 910–937.
45. Y. D. Sergeyev and V. A. Grishagin, Parallel asynchronous global search and the nested optimization scheme, *J. Comput. Anal. Appl.*, **3** (2001), 123–145.
46. Y. D. Sergeyev, R. G. Strongin and D. Lera, *Introduction to Global Optimization Exploiting Space-filling Curves*, Springer, 2013.
47. Y. D. Sergeyev, D. Famularo and P. Pugliese, Index branch-and-bound algorithm for Lipschitz univariate global optimization with multiextremal constraints, *J. Glob. Optim.*, **21** (2001), 317–341.
48. R. G. Strongin, *Numerical Methods in Multi-Extremal Problems (Information-Statistical Algorithms)*, Moscow: Nauka, In Russian, 1978.
49. R. G. Strongin, Algorithms for multi-extremal mathematical programming problems employing a set of joint space-filling curves, *J. Glob. Optim.*, **2** (1992), 357–378.
50. R. G. Strongin, V. P. Gergel, V. A. Grishagin and K. A. Barkalov, *Parallel Computations for Global Optimization Problems*, Moscow State University (In Russian), Moscow, 2013.
51. 5 (MR1797058) [10.1007/978-1-4615-4677-1] R. G. Strongin and Y. D. Sergeyev, *Global Optimization with Non-convex Constraints. Sequential and Parallel Algorithms*, Kluwer Academic Publishers, Dordrecht (2000, 2nd ed. 2013, 3rd ed. 2014).
52. A. Törn and A. Žilinskas, *Global Optimization*, Springer, 1989.
53. P. Venkataraman, *Applied Optimization with MATLAB Programming*, John Wiley & Sons, 2009.
54. A. A. Zhigljavsky, *Theory of Global Random Search*, Kluwer Academic Publishers, Dordrecht, 1991.

55. Gergel, V., Sidorov, S.: A Two-Level Parallel Global Search Algorithm for Solution of Computationally Intensive Multiextremal Optimization Problems. In: Malyshkin, V. (Ed.) PaCT 2015, LNCS, vol. 9251, pp. 505-515. Springer, Heidelberg (2015)
56. Strongin, R.G.: Parallel multi-extremal optimization using a set of evolvents. Comp. Math. Math. Phys. **31(8)**, 37–46 (1991)
57. Strongin, R.G.: Algorithms for multi-extremal mathematical programming problems employing the set of joint space-filling curves. J. Global Optim. **2(4)**, 357–378 (1992)
58. Strongin, R.G., Gergel, V.P., Barkalov, K.A.: Parallel methods for global optimization problem solving. Journal of instrument engineering. **52**, 25–33 (2009) (In Russian)
59. Gaviano, M., Kvasov, D.E, Lera, D., and Sergeyev, Ya.D.: Software for generation of classes of test functions with known local and global minima for global optimization. ACM Transactions on Mathematical Software 29(4), 469–480 (2003)
60. Grishagin, V.A.: Operating Characteristics of Some Global Search Algorithms. Problems of Statistical Optimization 7, 198–206 (1978) (In Russian)
61. Gergel V.P., Barkalov K.A., and Sysoyev A.V: Globalizer: A novel supercomputer software system for solving time-consuming global optimization problems. Numerical Algebra, Control & Optimization 8(1), 47–62, 2018