Comparison of dimensionality reduction schemes for parallel global optimization algorithms *

Konstantin Barkalov, Vladislav Sovrasov, and Ilya Lebedev

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

Abstract. This work considers a parallel algorithms for solving multiextremal 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

In the present paper, the parallel algorithms for solving the multiextremal optimization problems are considered. In the multiextremal problems, the opportunity of reliable estimate of the global optimum is based principally on the availability of some information on the function known *a* priori allowing relating the probable values of the optimized function to the known values at the points of performed trials. Very often, such an information on the problem being solved is represented in the form of suggestion that the objective function $\varphi(y)$ satisfies Lipschitz condition with the constant L not known a priori (see, for example, [18–20]). At that, the objective function could be defined by a program code i. e. could represent a "black-box"-function. Such problems are presented in the applications widely (problems of optimal design of objects and technological processes in various fields of technology, problems of model fitting according to observed data in scientific research, etc.).

Many methods destined to solving the problems of the class specified above reduce the solving of a multidimensional problem to solving the one-dimensional

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

subproblems implicitly (see, for example, the methods of diagonal partitions [14, 15] or simplicial partitions [16, 17]). In the present work, we will use the approach developed in Lobachevsky State University of Nizhni Novgorod based on the idea of the dimensionality reduction with the use of Peano space-filling curves $y(x)$ mapping the interval $[0, 1]$ of the real axis onto an n-dimensional cube continuously and unambiguously.

Several methods of constructing the evolvents approximating the theoretical Peano curve have been proposed in [2, 3, 7, 10]. These methods were implemented in the Globalizer software system [13]. The goal of the present study was comparing the properties of the evolvents and the selecting the most suitable ones for the use in the parallel global optimization algorithms.

2 Statement of Multidimensional Global Optimization Problem

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

$$\begin{gathered}\varphi(y^*) = \min\{\varphi(y) : y \in D\}, \\ D = \{y \in \mathbb{R}^N : a_i \le y_i \le b_i, 1 \le i \le N\}\end{gathered} \tag{1}$$

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

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

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

Usually, the objective 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 time-consuming.

3 Methods of Dimension Reduction

3.1 Single evolvent

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 [1], [2], [4], [5].

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 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 [5].

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 Shifted evolvents

One of the possible ways to overcome the negative effects of using a numerical approximation of evolvent (it destroys the information about the neighbour points in $\mathbb{R}^N$ space, see [3]) 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 [3, 5, 9]).

Such set of evolvents can be produced by shifting the source evolvent $y^0(x)$ by $2^{-l}, 0 \leq l \leq L$ on each coordinate. Each evolvent has it's own corresponding hypercube $D_l = \left\{y \in R^N : -2^{-1} \leq y_i + 2^{-l} \leq 3 \cdot 2^{-1},\ 1 \leq i \leq N\right\},\ 0 \leq l \leq L$.

In Fig. 1a 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 D_l, having introduced an additional constraint function

$$g_0(y) = \max\left\{|y_i| - 2^{-1} :\ 1 \leq i \leq N\right\}, \tag{6}$$

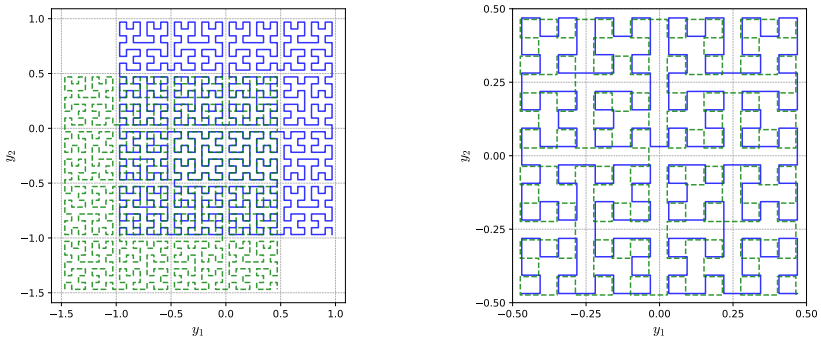
one can present the initial hypercube D in the form

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

i.e., $g_0(y) \leq 0$ 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 $y^l(x),\ 0 \leq l \leq L$.

Thus, each evolvent $y^l(x),\ 0 \leq l \leq L$, 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 (6)

$$\min\left\{\varphi(y^l(x)) : x \in [0,1],\ g_j(y^l(x)) \leq 0,\ 0 \leq j \leq m\right\},\ 0 \leq l \leq L. \tag{7}$$



(a) Two shifted evolvents on the hypercubes D_0 and D_1

(b) Two rotated evolvents on the same plane

Fig. 1: Multiple evolvents built with low density

3.3 Rotated evolvents

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 (see the end of the section 5.1).

To overcome complexity of the S-evolvent and to preserve the information on the nearness of the points in the N-dimensional space, one more scheme of building of the multiple mappings was 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 [10]. In Fig. 1b 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 (6), 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.

3.4 Non-Univalent evolvent

As it has been already mentioned above (Sec. 3.2), the loss of information on the proximity of the points in the multidimensional space could be compensated in part by the use of multiple mappings $Y_L(x) = \{y^1(x), ..., y^L(x)\}$. However, the Peano-type curve preserves a part of this information itself: it is not an injective mapping. Therefore, if a single image $y(x) \in \mathbb{R}^N$ is available, one can obtain

several different preimages $t_j \in [0,1], t_j \neq x$, which could be added into the search information of the method later.

The Peano-type curve used in (3) for the dimensionality reduction is defined via the transition to the limit. Therefore, it cannot be computed directly. In the numerical optimization, some approximation of this curve is used, and it is an injective piecewise-linear curve. In [2] a non-univalent mapping of a uniform grid in the interval $[0,1]$ onto a uniform grid in a hypercube D has been proposed. Each multidimensional node can have up to 2^N one-dimensional preimages. In Fig. 2b, the grid in the $\mathbb{R}^2$ space is marked by the crosses, for two nodes of which the corresponding one-dimensional preimages from $[0,1]$ are pointed (marked by the squares and circles). Each node mentioned above has 3 preimages.

A potentially large number of preimages (up to 2^N) and the inability to use the parallel scheme for the multiple mappings form Sec. 4.2 are the disadvantages of the non-univalent evolvent.

3.5 Smooth evolvent

The methods of constructing the evolvents considered in the previous paragraphs produce the curve $y(x)$, which in not a smooth one (see Fig. 1a). The absence of smoothness may affect the properties of the reduced one-dimensional function $\varphi(y(x))$ adversely since a smooth curve reflect the information on the growth/decay of the initial function better. On the basis of initial algorithm of constructing the non-smooth evolvent, a generalized algorithm allowing constructing a smooth space-filling curve has been proposed [7]. As an illustration, a smooth evolvent for the two-dimensional case is presented in Fig. 2a. An increased computational complexity (several times as compared to the piecewise-linear curves) is a disadvantage of the smooth evolvent. This caused by computing of the nonlinear smooth functions.

4 Parallel Computations for Solving Global Optimization Problems.

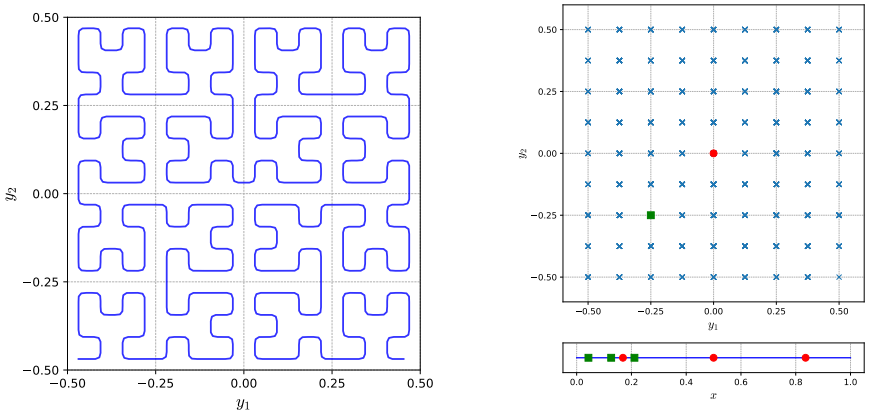
4.1 Core Multidimensional Algorithm of Global Search (MAGS)

The optimization methods applied in Globalizer to solve the reduced problem (3) are based on the MAGS method, which can be presented as follows — see [2], [5].

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 $0 = x_0 < x_1 < ... < x_{k+1} = 1$.

Rule 2. Compute the characteristics $R(i)$ for each interval $(x_{i-1}, x_i), 1 \leq i \leq k+1$.



(a) Smooth evolvent (b) Non-univalent evolvent

Fig. 2: Different evolvents built with low density

Rule 4. Determine the interval with the maximum characteristic $R(t) = \max_{1 \leq i \leq k+1} R(i)$.

Rule 5. Execute a new trial at the point x^{k+1} located within the interval with the maximum characteristic from the previous step $x^{k+1} = d(x_t)$.

The stopping condition, which terminated the trials, is defined by the inequality $\rho_t < \varepsilon$ for the interval with the maximum characteristic from Step 4 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.

The convergence conditions and exact formulas for descision rules $R(i)$ and $d(x)$ of the described algorithm are given, for example, in [5].

4.2 Parallel algorithm exploiting a set of evolvents

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

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

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 point y^k (the blocking of the point y^k).

3. Determine the preimages $x^{k_s} \in [0,1], 1 \leqslant s \leqslant S$, 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 $x^{(s+iS)}, 0 \leqslant i < p$. 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 $y^{(s+iS)}, 0 \leqslant i < p, 0 \leqslant s < S$ is performed (from each process to each one). After that, the preimages $x^{(q+iS)}, 0 \leqslant q < S, q \neq s$ 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 Nizhni 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). All considered algorithms and evolvents were implemented using C++ within the Globalizer software system [13]. In order to enable the parallelism, OpenMP was used on a single node, and MPI was used for the parallelization on several nodes.

The comparison of the global optimization algorithms was performed by the evaluation of the quality of solving a set of problems from some test class. In the present paper, the test class generated by GKLS (Gaviano, Kvasov, Lera, Sergeyev) generator [11] was considered. The generator creates objective functions by distorting a convex quadratic function by polynomials in order to introduce local minima. Thus, GKLS allows constructing the complex multiextremal problems of various dimensions. In the present work, the series of 100 problems from the classes of the dimensions of 2, 3, 4, and 5 were considered. Each class had two degrees of complexity — *Simple* and *Hard*. These the classes have different radius of the attraction region of the global minimizer (*Hard* has smaller region) and distance from the global minimizer to the vertex of the quadratic function (*Simple* has smaller distance). The parameters of the generator for the considered classes were given in Ref. [11].

In order to evaluate the efficiency of an algorithm on a given set of 100 problems, we will use the operating characteristics [12], which are defined as a curve, showing the dependency of number of solved problems vs the number of iterations.

5.1 Comparison of the sequential evolvents

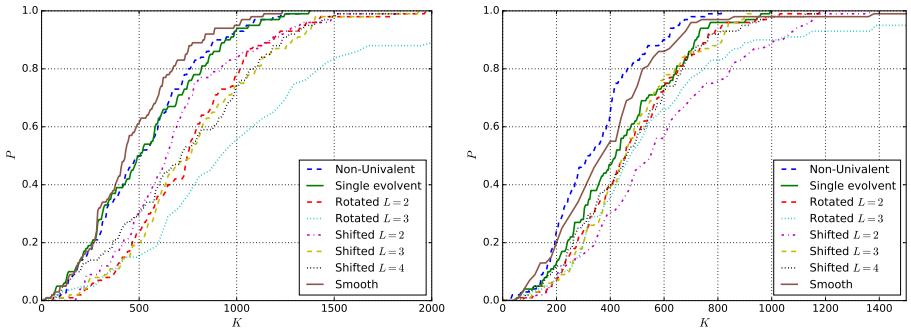
In order to understand whether any type of evolvents listed above has an essential advantage as compared to other ones, the operating characteristics of the index method with different types of evolvents have been obtained for the classes GKLS 2d Simple and GKLS 3d Simple. The global minimum was considered 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 $\delta = 0.01$.

In all experiments, the evolvent construction density parameter $m = 12$. The minimum value of the reliability parameter r was found for each type of evolvents by scanning over a uniform grid with the step 0.1.

On the GKLS 2d Simple class at the minimum r, the non-univalent evolvent and the smooth one provide a faster convergence (Fig. 3b). The same was observed at $r = 5.0$ as well (Fig. 3a). In the latter case, the shifted evolvent and the rotating one begin to lag behind the rest since the value $r = 5.0$ is too big for them.

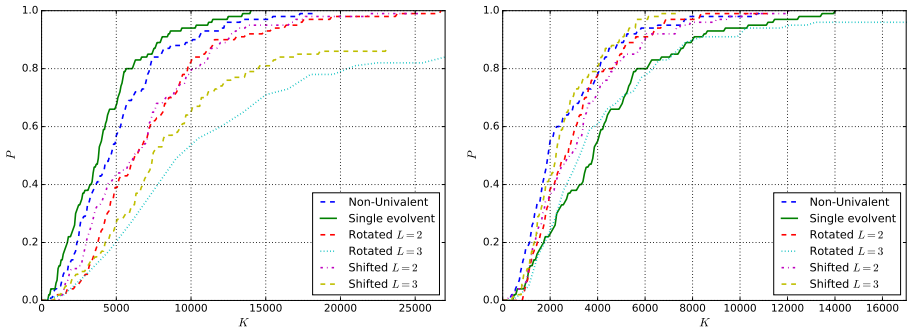
On the GKLS 2d Simple class at the minimum r, the non-univalent evolvents and multiple ones have a considerable advantage over the single evolvent (Fig. 4b). The value $r = 4.5$ is too big for the rotated evolvents and for the shifted one (Fig. 4a).

Overhead costs when using the shifted evolvents. In all experiments presented above, the number of computations of the objective function from the GKLS class was taken into account when plotting the operating characteristics. However, in the case of the shifted evolvent, the index method solves the problem with the constraint g_0 from (6). At the points where g_0 is violated, the value of the objective function is not computed. Nevertheless, these points are stored in the



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

Fig. 3: Operating characteristics on GKLS 2d Simple class



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

Fig. 4: Operating characteristics on GKLS 3d Simple class

search information producing the additional computational costs. In Table 1, the averaged numbers of calls to g_0 and to the objective function are presented. At $L = 3$, the constraint g_0 was computed almost 20 times more than the objective function φ i. e. 95% of the whole search information account for the auxiliary points. Such overhead costs are acceptable when solving the problems of small dimension with the computation costly objective functions. However, when increasing dimensionality and total number of trials other types of evolvents are preferred.

5.2 Parallel rotated evolvents

In order to evaluate the efficiency of the parallel algorithm from Sec. 4.2, the numerical experiments on the GKLS 4d (Hard, Simple) classes and on the GKLS 5d (Hard, Simple) ones were conducted. The value of r in all experiments was equal to 5.0, the size of the δ-vicinity of the known solution was increased up

Table 1: Averaged number of computations of g_0 and of φ when solving the problems from GKLS 3d Simple class using the shifted evolvent

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

to 0.3. When solving the series of problems, up to 8 cluster nodes and up to 32 computational threads on each node were employed.

In Table 2, an averaged number of iterations when solving 100 problems from each considered class is presented. The number of iterations is reduced considerably with increasing the number of nodes and the number of threads on each node (except the GKLS 4d Simple class at the transition from 1 node to 4 ones in the single thread mode).

Table 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

If one assumes the costs of parallelization to be negligible as compared to the costs of computing the objective functions in the optimization problems, the speedup in time due to the use of the parallel method would be equal to the speedup with respect to the number of iterations. However, actually this suggestion is not always true. In all numerical experiment the time of computing the objective function was approximately 10^{-3} sec. In Table 3, the speedups in iterations and in time (in the pendent brackets) are presented. In the first row of the table corresponding to the sequential mode, the averaged time of solving a single problem is presented in the pendent brackets. One can see from the table that for the GKLS 4d classes it is more efficient to utilize a single node in the multithread mode whereas for solving more complex five-dimensional problems, the use of several nodes is better, each node is operating in the parallel mode.

Table 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 Conclusions

In the present work, 5 different Peano curve-type mappings applied to the dimensionality reduction in the global optimization problems were considered. From the preliminary comparison conducted in Sec. 5.1, one can make the following conclusions:

- the smooth evolvent and the non-univalent one demonstrate the best result in the problems of small dimensionality and can be applied successfully in solving the problems with the computational costly objective functions. The properties of these evolvents don't allow developing the optimization algorithms scalable onto several cluster nodes based on these ones.
- the shifted evolvents introduce large overhead costs on the operation of the method due to the requirement to adding an auxiliary functional constraint into the problem (1). The experiments have demonstrated that up to 95% of the search information account for the points, in which the auxiliary constraint is computed only. The shifted evolvents can be used as the base for the parallel algorithm from Sec. 4.2. However, the costs of processing the auxiliary points would result likely in a small speedup from the parallelization. However, if the objective function is computation-costly enough, the use of these evolvents could make sense.
- the rotated evolvents have provided an acceprable speed of convergence in the problems of small dimensionality in the sequential mode. The use of these ones don't result in the introduction of the auxiliary constraints that allows constructing an efficient parallel algorithm based on these evolvents.

In Sec. 5.2 the results of the numerical experiments are presented, which have demonstrated the algorithm from Sec. 4.2 based on the rotated evolvents allowed obtaining the speedup up to 43 times when solving the problem series employing several nodes of the computer cluster. It is worth noting that the objective functions in the considered problems are not computation-costly (the averaged computation time was 10^{-3} sec). In the case of more complex problems,the speedup in time could approach the speedup with respect to the number of iterations.

References

1. Y. D. Sergeyev, R. G. Strongin and D. Lera, *Introduction to Global Optimization Exploiting Space-filling Curves*, Springer (2013)
2. R. G. Strongin, *Numerical Methods in Multi-Extremal Problems (Information-Statistical Algorithms)*, Moscow: Nauka (1978) (In Russian)
3. R. G. Strongin, Algorithms for multi-extremal mathematical programming problems employing a set of joint space-filling curves, *J. Glob. Optim.*, **2**, 357–378 (1992)
4. R. G. Strongin, V. P. Gergel, V. A. Grishagin and K. A. Barkalov, *Parallel Computations for Global Optimization Problems*, Moscow State University, Moscow (2013) (In Russian)
5. 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)
6. A. Törn and A. Žilinskas, *Global Optimization*, Springer, (1989)
7. Goryachih, A. *A class of smooth modification of space-filling curves for global optimization problems* Springer Proceedings in Mathematics and Statistics, **197**, pp. 57–65 (2017)
8. A. A. Zhigljavsky, *Theory of Global Random Search*, Kluwer Academic Publishers, Dordrecht (1991)
9. Strongin, R.G.: *Parallel multi-extremal optimization using a set of evolvents.* Comp. Math. Math. Phys. **31(8)**, 37–46 (1991)
10. 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)
11. 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)
12. Grishagin, V.A.: *Operating Characteristics of Some Global Search Algorithms.* Problems of Statistical Optimization **7**, 198–206 (1978) (In Russian)
13. 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)
14. Sergeyev, Ya.D., Kvasov, D.E.: *Global search based on efficient diagonal partitions and a set of Lipschitz constants.* SIAM J. Optim **16(3)**, 910–937 (2006)
15. Sergeyev, Y.D., Kvasov, D.E.: *A deterministic global optimization using smooth diagonal auxiliary functions.* Communications in Nonlinear Science and Numerical Simulation. **21(1-3)**, 99–111 (2015)
16. Žilinskas, J.: *Branch and bound with simplicial partitions for global optimization.* Math. Model. Anal. **13(1)**, 145–159 (2008)
17. Paulavičius, R., Žilinskas, J.: *Simplicial Lipschitz optimization without the Lipschitz constant.* J. Glob. Optim. **59(1)**, 23-40 (2014)
18. Jones, D.R.: The direct global optimization algorithm. In: Floudas, C.A., Pardalos, P.M. (eds.) *The Encyclopedia of Optimization*, 2nd edn., pp. 725–735. Springer, Heidelberg (2009)
19. Gablonsky, J.M., Kelley, C.T.: *A locally-biased form of the DIRECT algorithm.* J. Glob. Optim., **21(1)**, 27–37 (2001)
20. Evtushenko, Y., Posypkin, M.: *A deterministic approach to global box-constrained optimization.* Optim. Lett. **7(4)**, 819–829 (2013)