Manuscript submitted to
AIMS' Journals
Volume **X**, Number **0X**, XX **200X**

doi:10.3934/xx.xx.xx.xx

pp. **X–XX**

GLOBALIZER: A NOVEL SUPERCOMPUTER SOFTWARE SYSTEM FOR SOLVING TIME-CONSUMING GLOBAL OPTIMIZATION PROBLEMS

Victor Gergel, Konstantin Barkalov* and Alexander Sysoyev

Lobachevsky State University of Nizhny Novgorod
23, Gagarin Avenue
Nizhny Novgorod, Russia

(Communicated by the associate editor name)

Abstract. In this paper, we describe the Globalizer software system for solving the global optimization problems. The system is designed to maximize the use of computational potential of the modern high-performance computational systems in order to solve the most time-consuming optimization problems. The highly parallel computations are facilitated using various distinctive computational schemes: processing several optimization iterations simultaneously, reducing multidimensional optimization problems using multiple Peano space-filling curves, and multi-stage computing based on the nested block reduction schemes. These novelties provide for the use of the supercomputer system capabilities with shared and distributed memory and with large numbers of processors to solve the global optimization problems efficiently.

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 criterion to be optimized has several local optima within the search domain, which have different values. The existence of several local optima essentially 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 supercomputer systems only.

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

2010 *Mathematics Subject Classification.* Primary: 90C26; Secondary: 65Y05.

Key words and phrases. Global optimization, information-statistical theory, parallel computing, high-performance computer systems, supercomputer technologies.

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

* Corresponding author: Konstantin Barkalov.

systems 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 the market of the global optimization software, one can select from the following systems:

- LGO (Lipschitz Global Optimization) [38] was designed to solve the global optimization problems, where the criteria and constraints satisfy the Lipschitz condition. The system is a commercial product based on the diagonal extensions of the one-dimensional multiextremal optimization algorithms.
- GlobSol [27] is oriented on solving the global optimization problems as well as of the systems of the nonlinear equations. The system includes the interval methods based on the branch and bound approach. There are some extensions of the system for the parallel computations, and it is available for free usage.
- LINDO [32] is featured by a wide spectrum of optimization methods for solving linear, integer, stochastic, nonlinear, and global optimization problems. The ability to interact with the Microsoft Excel software environment is a useful feature of this system. The system is widely used in the practical applications and is available for free usage.
- IOSO (Indirect Optimization on the basis of Self-Organization) [9] is oriented on solving a wide class of the optimization problems including the global optimization ones. The system is widely used to solve the applied problems in various application fields. There is a version of the system for exciting on the parallel computational systems. The system is a commercial product, but it is available for the trial usage.
- MATLAB Global Optimization Toolkit [53] contains a wide spectrum of methods for solving the global optimization problems, including multistart algorithms, global pattern search, simulated annealing methods, etc. The library is compatible with the TOMLAB system [24], which is an additional MATLAB extension. It should also be noted that similar libraries for solving the global optimization problems are available for MathCAD, Mathematica, and Maple systems as well.
- BARON (Branch-And-Reduce Optimization Navigator) [41] was designed to solve continuous integer programming and global optimization problems using the branch and bound approach. BARON is included in the GAMS (General Algebraic Modeling System) system [6].

- Global Optimization Library in R is a large collection of optimization methods implemented in the R language [36]. Among these methods, stochastic and deterministic global optimization algorithms and the branch and bound method were implemeted.

The list provided above is certainly not exhaustive – additional information on the software systems for solving a wider spectrum of optimization problems can be found, for example, in [35], [39], [40].

In this paper, a novel Globalizer software system is presented. Globalizer expands the collection of the parallel optimization systems mentioned above. The Globalizer system was designed to solve the time-consuming multiextremal optimization problems that is an advantage of the system. In order to obtain the global optimized solutions within a reasonable time, the system exploits the huge computational potential of high-performance massively parallel systems.

The further structure of the paper is as follows. In Section 2, a statement of the multidimensional global optimization problem is given and an approach for reducing this one to a one-dimensional optimization problem is described. In Section 3, the parallel computation schemes are discussed and the parallel optimization methods are presented. In Section 4, the architecture of the Globalizer system is described. In Section 5, the results of the numerical experiments, which confirm the parallel efficiency of the Globalizer system, are presented. Finally, Section 6 presents the conclusions and some ideas for the future research.

2. Multidimensional Global Optimization Problems and Dimension Reduction.

In this paper, the core class[1] 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:

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

It is supposed, that the objective function $\varphi(y)$ satisfies the Lipschitz condition

$$|\varphi(y_1) - \varphi(y_2)| \le L\|y_1 - y_2\|, y_1, y_2 \in D, \qquad (2)$$

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 base 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

[1] In general, the Globalizer can be applied for solving multicriterial multiextremal multidimensional optimization problems with nonlinear constraints.

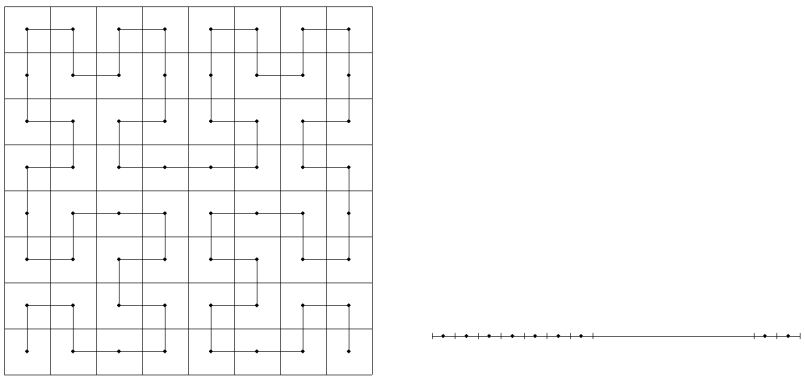

FIGURE 1. A Peano curve approximation for the third level

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 for 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].

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 satisfy the uniform Hölder condition i. e.

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

where the constant H is defined by the relation $H = 4L\sqrt{N}$, 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 presented 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.

The computational scheme obtained as a result of the dimensionality reduction consists of the following (see Figure 2):

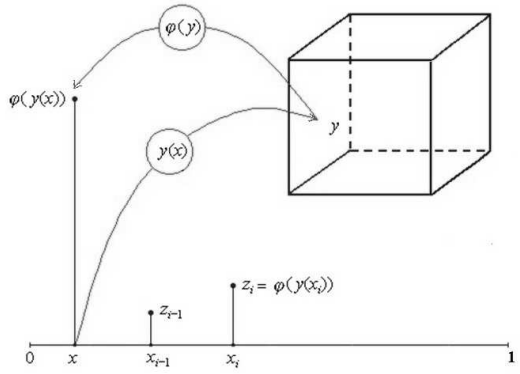

FIGURE 2. The computational scheme for obtaining the value of the reduced one-dimensional function $\varphi(y(x))$

- 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 in the mapping $y(x)$ is calculated,
- The value of the initial multidimensional function $\varphi(y)$ is calculated at the multidimensional 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. **Parallel Computations for Solving Global Optimization Problems.** Let us consider the parallelization approaches used widely in the theoretical and practical applications of parallel computing within the context of global optimization problems:

- The distribution of the search domain D among the available computational units (data parallelization scheme). This approach is insufficient in the case of optimization problems since in such computations the subdomain, which contain the sought global minimum, will only be processed by one processor, and, therefore, all the remaining processors would perform the excess computations.
- The parallelization of the optimization algorithms (task parallelization scheme). This approach is also insufficient since the direct computational costs for executing the optimization algorithms are relatively low (the majority of computations are represented by the calculations of the optimized function values due to the initial assumption of considerable computational costs of such calculations).
- The parallelization of the computations executed in order to obtain the values of the optimized function. This approach is applicable since the most time-consuming part of the global optimization will be parallelized. However, this method is not featured by the generality (the development of parallelization methods is to be performed every time from the very beginning while solving each particular optimization problem).

Within the framework of the information-statistical theory, a general parallelization approach for solving the global optimization problems has been proposed [[50], [51] — *the parallel computations are provided by means of simultaneous computing the values of the minimized function $\varphi(y)$ at several different points within the search domain D*. This approach provides the parallelization for the most time-consuming part of the optimization computations.

The global optimization algorithms implemented in Globalizer will be described step-by-step. In Subsection 3.1, the core serial Multidimensional Algorithm of Global Search (MAGS) is presented. In Subsection 3.2, a parallel generalization of the MAGS algorithm for the parallel computations on the multiprocessor computational nodes with shared memory is described. In Subsection 3.3, the scheme for the parallel computations on the high-performance computational systems with distributed memory is given.

3.1. **Core Multidimensional Generalized 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], [30], [18], [19], [23], [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

$$f(x) = \varphi(y(x)) : x \in [0,1]. \tag{5}$$

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

$$0 = x_0 < x_1 < ... < x_{k+1} = 1. \tag{6}$$

the points x_0, x_k were introduced additionally for the convenience of further explanation, the values of the minimized function z_0, z_k at these points are undefined.

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

$$m = \max_{1 \leq i \leq k} \frac{|z_i - z_{i-1}|}{\rho_i}, \; M = \begin{cases} rm, & m > 0, \\ 1, & m = 0, \end{cases} \tag{7}$$

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 (6). 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

$$R(i) = 2\rho_i - 4\frac{z_i}{M}, \quad i = 1,$$

$$R(i) = \rho_i + \frac{(z_i - z_{i-1})^2}{M^2 \rho_i} - 2\frac{z_i + z_{i-1}}{M}, \quad 1 < i < k+1, \tag{8}$$

$$R(i) = 2\rho_i - 4\frac{z_{i-1}}{M}, \quad i = k+1.$$

Rule 4. Determine the interval with the maximum characteristic

$$R(t) = \max_{1 \le i \le k+1} R(i). \tag{9}$$

Rule 5. Execute a new trial at the point x^{k+1} located within the interval with the maximum characteristic from (9)

$$x_{k+1} = \frac{x_t + x_{t-1}}{2} - \operatorname{sign}(z_t - z_{t-1}) \frac{1}{2r} \left[\frac{|z_t - z_{t-1}|}{M} \right]^N, \quad \text{if } 1 < t < k+1, \tag{10}$$

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

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

$$\rho_t < \varepsilon \tag{11}$$

for the interval with the maximum characteristic from (9) and $\varepsilon > 0$ is the predefined accuracy of the optimization problem solution. If the stop condition is not satisfied, the index k is incremented by 1, and new global optimization iteration is executed.

In order to explain the algorithm presented above, let us note the following. The characteristics $R(i), 1 \le i \le k+1$, calculated according to (8) 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 (9) and (10) for selecting the interval for executing 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 for the described algorithm are given, for example, in [51].

3.2. **Parallel Computations for Systems with Shared Memory.** Modern supercomputer computational systems consist of plenty of computational nodes, which include several multicore processors. In this case, random access memory is shared among the computational nodes — the content of any memory location can be read (written) by any computational core at any arbitrary moment. In most cases, shared memory is uniform — the time characteristics for accessing the memory are the same for all computational cores and for all memory locations.

The following approach can be applied to provide the parallel computations with the MAGS method. As it was mentioned above, the characteristics $R(i), 1 \le i \le k+1$ calculated according to (8) can be interpreted as some measures of importance of the intervals with respect to the expected location of the global minimum point. Following this understanding, every next global optimization iteration is executed within the most important interval with the maximum characteristic. As a result, it becomes evident how to select the other intervals for simultaneous computations of the minimized function values at several different points within the search domain — these ones should be the intervals with the next magnitudes of characteristics.

The computational scheme of the Parallel Multidimensional Algorithm of Global Search (PMAGS) for the computational systems with shared memory can be developed based on the approach described above. This scheme is almost identical to the MAGS one — the differences consist just in the following set of rules.

Rule 4'. Arrange the characteristics of the intervals computed according to (8) in the decreasing order

$$R(t_1) \ge R(t_2) \ge \cdots \ge R(t_k) \ge R(t_{k+1}) \tag{12}$$

and select p intervals with the indices $t_j, 1 \le j \le p$, with the maximum values of characteristics (p is the number of processors/cores used for the parallel computations).

Rule 5'. Execute new trials at the points $x_{k+j}, 1 \le j \le p$ located in the intervals with the maximum characteristics from (12)

$$x_{k+j} = \frac{x_{t_j} + x_{t_j-1}}{2} - \mathrm{sign}(z_{t_j} - z_{t_j-1})\frac{1}{2r}\left[\frac{|z_{t_j} - z_{t_j-1}|}{M}\right]^N, \quad \text{if } 1 < t_j < k+1, \tag{13}$$

$$x_{k+j} = \frac{x_{t_j} + x_{t_j-1}}{2}, \quad \text{if } t_j = 1 \text{ or } t_j = k+1.$$

The stop condition (11) should be checked for all intervals from (12), where the scheduled trials are executed

$$\rho_{t_j} < \varepsilon, 1 \le j \le p. \tag{14}$$

If the stop condition is satisfied at least for one interval $t_j, 1 \le j \le p$, the calculations are terminated. Otherwise, the index k is incremented by p, and new global optimization iteration is executed.

The convergence conditions for the developed parallel algorithm are considered in [50], [51]. Thus, in particular, when the convergence conditions are satisfied, the parallel computations are non-redundant as compared to the serial method when using up to 2^N processors/cores (N is the dimensionality of the global optimization problem).

3.3. **Parallel Computations for Systems with Distributed Memory.** The next level of parallel computations on the high-performance computational systems consists in using several computational nodes. In this case, each computational node has its own separate memory and the interaction between different nodes can be provided by means of data transfer via the computer communication network.

To provide parallel computations on high-performance computational systems with the distributed memory, it is proposed to use a set of mappings

$$Y_s(x) = \{y^1(x), \ldots, y^s(x)\} \tag{15}$$

instead of applying a single Peano curve $y(x)$ — see, for instance, [49], [50], [51]. In order to construct the set $Y_s(x)$, several different approaches can be applied. Thus, for example, in [49] each mapping $y^i(x)$ from $Y_s(x)$ is constructed as the result of some shift along the main diagonal of the hypercube D. This way, the set of the constructed Peano curves enables the close prototypes x', x'' to be obtained from the interval $[0, 1]$ for any close multidimensional images y', y'' from D differing in one coordinate only.

Some other methods for constructing the multiple mappings were considered in [50].

The set of mappings $Y_s(x)$ from (15) generates s information-linked one-dimensional problems of the kind (3):

$$\varphi(y^l(x^*)) = \min\{\varphi(y^l(x)) : x \in [0, 1]\}, 1 \le l \le s. \tag{16}$$

It is important to note that the family of the one-dimensional problems $\varphi(y^l(x)), 1 \le l \le s$ obtained as a result of the dimensionality reduction is an information-linked one — the function values computed for any problem $\varphi(y^l(x))$ from the family (16) can be used for all rest problems of this family.

The information compatibility of the optimization problems from the family (16) allows executing the parallel computations in the following way. Each particular problem can be solved by a separate processor of the computational system; the exchange of search information between the processors should be performed during the computations. As a result, a unified approach for the parallel computations on the high-performance computational systems with distributed and shared memory can be developed. This method consists in the following.

1. The family of the reduced one-dimensional information-linked optimization problems (16) is distributed among the computational nodes of the multi-node computational system.
2. Each optimization problem from the family (16) is solved on the corresponding computational node using the PMAGS method (described above in Subsection 3.2) supplemented by the following rules of the information interaction.
   (a) Prior to beginning new trial at any point $x' \in [0,1]$ for any problem $\varphi(y^l(x)), 1 \le l \le s$, the following should be performed:
   - compute the image $y' \in D$ for the point $x' \in [0,1]$ according to the mapping $y^l(x)$,
   - determine the preimages[2]
   $$x'_i : y' = y^i(x'_i), 1 \le i \le s,$$
   in accordance with the set of mappings $y^l(x), 1 \le l \le s$ for each problem from the family (16),
   - send the prototypes $x'_i, 1 \le i \le s$ to all computational nodes in order to exclude the repeated selection of the intervals, which the prototypes fall in, for using these ones to determine the points of new trials. To perform the data transfer, a queue can be created at each computational node for sending the trial points and receiving the minimized function values at these points.

   (b) After completing any trial for any problem $\varphi(y_l(x)), 1 \le l \le s$, at any point $x' \in [0,1]$, it is necessary:
   - to determine all prototypes $x'_i, 1 \le i \le s$ for the point of the completed trial for each problem from the family (16),
   - to send the prototypes $x'_i, 1 \le i \le s$ and the result of the trial $z' = \varphi(y_l(x'))$ to all computational nodes to include the obtained data into the search information processed according to the rules of the parallel global optimization algorithm.

   (c) Prior to starting the next global optimization iteration, the algorithm should check the queue of the received data; if there are any data in the queue, these should be included into the search information. The MAGS algorithm uses these accumulated data for the adaptive computing of the optimization iteration points (see Section 3.1).

The possibility of asynchronous data transfer (the computational nodes process the received data upon receipt only) is an important feature of the proposed parallel computation scheme. In addition, there is no any selected managing node within this scheme. The number of computational nodes can vary during the global optimization, and excluding any node does not result in the loss of the sought global minimum of the minimized multiextremal function.

[2]All preimages have the same image y'; the preimages are calculated by the inverse mappings $(y^i)^{-1}$, $1 \le i \le s$; the number of the preimages coincides with the number of the mappings.

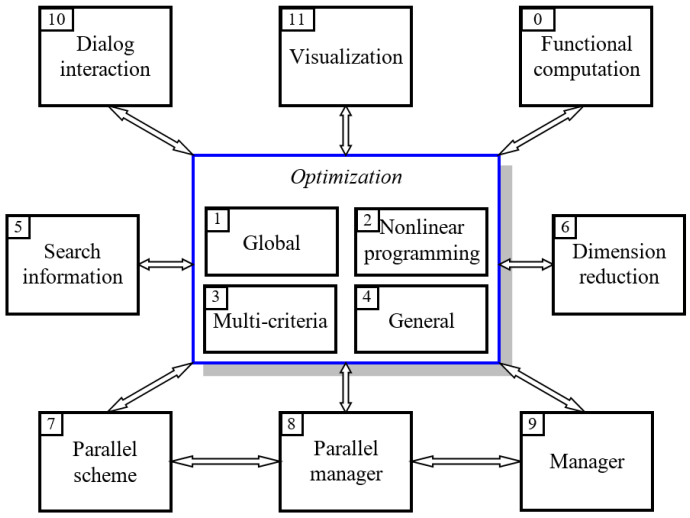

FIGURE 3. The architecture of the Globalizer system

This computational scheme generates the Generalized Parallel Multidimensional Algorithm of Global Search (GPMAGS) for high-performance computational systems, which may include many multiprocessor (multicore) computational nodes with distributed memory as well as the computational accelerators based on Intel Xeon Phi processors and on the general purpose graphic processors (GPUs).

Additional information on such parallel computation schemes can be found in [22].

4. **Globalizer System Architecture.** Globalizer expands the family of the global optimization software systems developed by the authors during the past several years. One of the first developments was the SYMOP multiextremal optimization system [17], which has been applied for solving many optimization problems. The ExaMin system [2], [3], [4], [16] was developed and used to investigate various parallel algorithms for solving the global optimization problems on the high-performance computational systems.

The architecture of the Globalizer system is presented in Figure 3. The system components are:

- Block 0 consists of the procedures for computing the function values (criteria and constraints) of the optimization problem being solved; it is an external block with a predefined interface with respect to the Globalizer system and is provided by the user.
- Blocks 1-4 form the optimization subsystem and solve the global optimization problems (Block 1), nonlinear programming ones (Block 2), multicriterial optimization problems (Block 3), and general decision making problems (Block 4). The interaction between these components is organized in a hierarchical order — the decision making problems are solved using the multicriterial optimization block, which, in turn, uses the nonlinear programming block, and so on.

- Block 5 is a subsystem for accumulating and processing the search information; it is one of the main subsystems – the size of search information may appear to be quite large and the efficiency of the global optimization methods depends on how completely the available search data are utilized.
- Block 6 contains the dimensionality reduction procedures based on the Peano space-filling curves; the reduced (one-dimensional) optimization problems are solved by the optimization blocks. This block also provides the interaction between the optimization blocks and the multidimensional optimization problem being solved.
- Block 7 provides tuning the parallel computation schemes in the Globalizer system subject to the employed computational system architecture (the number of the computational cores, the availability of shared and/or distributed memory, the availability of computational accelerators, etc.) and the applied global optimization methods.
- Block 8 is responsible for managing the parallel processes when performing the global optimization (determining the optimal configuration of parallel processes, distributing the processes between the computational unit, balancing the computational loads, etc.).
- Block 9 is a managing subsystem, which controls the parallel computations when solving the global optimization problems.
- Block 10 provides the dialog interaction with the users for stating the optimization problem, adjusting the system parameters (if necessary), and visualizing and presenting the global optimization results.
- Block 11 is a set of tools for visualizing and presenting the global optimization results; the availability of the tools for visual demonstration of the computational results enables the user to perform an efficient control over the global optimization process.

The Globalizer system comes in two different versions:

- Globalizer Pro is a research version for making various large-scale numerical experiments on the high-performance computational systems to investigate the new parallel global optimization methods being developed.
- Globalizer Lite is a limited version for solving the global optimization problems with a medium complexity level, and is accessible free of charge.

5. **Results of Numerical Experiments.** The numerical experiments were carried out using the Lobachevsky supercomputer at the State University of Nizhni Novgorod (the operation system – CentOS 6.4, the supercomputer management system – SLURM). Each supercomputer node had 2 Intel Sandy Bridge E5-2660 processors, 2.2 GHz, with 64 GB RAM. Each processor had 8 cores (i. e. total 16 CPU cores were available at each node). In order to generate the executable program code, the Intel C++ 14.0.2 compiler was used.

A number of computational nodes had Intel Xeon Phi 5110P processors, each containing 60 cores (total 240 threads). In addition, the supercomputer included the computational nodes equipped with two NVIDIA Kepler K20X GPUs, each providing 14 stream multiprocessors (total 2688 CUDA-cores). In order to generate the executable code, the CUDA Toolkit 6.0 was used.

In the performed experiments the optimization problems generated by the GKLS-generator [15] were used. This generator allows obtaining the multiextremal optimization problems with *a priori* known properties: the number of local minima,

TABLE 1. Averaged number of executed iterations of the compared global optimization methods

N	Problem class	DIRECT	DIRECTl	GSA
4	*Simple*	>47282(4)	18983	11953
	Hard	>95708(7)	68754	25263
5	*Simple*	>16057(1)	16758	15920
	Hard	>217215(16)	>269064(4)	>148342(4)

the sizes of the attractors of these ones, the global minimum point, the function value at this point, etc. In order to simulate the computational costs inherent to the applied optimization problems, in all experiments computing the objective function was intentionally complicated by the auxiliary computations not altering the form of the function and the position of its minima.

The numerical experiments were carried out first using the ExaMin system and then were reproduced using the Globalizer system. In all the executed experiments, 100 global optimization problems generated randomly by the GKLS-generator were solved. The results of experiments were averaged over the number of the solved optimization problems.

5.1. **Comparison with over methods.** The results of numerical comparison of three sequential algorithms — DIRECT [26], DIRECTl [14], and MAGS from Section 3.1 — are presented below.

The numerical comparison was carried out by using the function classes Simple and Hard of the dimensions 4 and 5. The results of numerical experiments for DIRECT and DIRECTl are taken from [44]. According to [44], the global minimizer y^* was considered to be found if the algorithm generated a trial point y^k inside a hyperinterval with a vertex y^* and

$$|y^k(i) - y^*(i)| \leq \sqrt[N]{\Delta}(b(i) - a(i)), i \leq i \leq N$$

where N is the problem dimensionality, a and b are the borders of the search domain D [3].

The values of the parameter $\Delta = 10^{-6}$ at $N = 4$ and $\Delta = 10^{-7}$ at $N = 5$ were used. When using MAGS, the reliability parameter was selected as follows: $r = 4.5$ for the Simple class and $r = 5.6$ for the Hard class. The density level parameter of the Peano curve approximation was fixed as $m = 10$.

The averaged numbers of iterations k_{av} executed by the global optimization methods for solving the optimization problems are shown in Table 1. The symbol ">" denotes the situation, when not all problems of the class are solved by the optimization method. This means that the algorithm was stopped as the maximum allowed number of iterations K_{max} was achieved. In this case, the value $K_{max} = 10^6$ was used to calculate the averaged number of iterations kav that corresponds to the lower estimate of this averaged value. The number of the unsolved problems is specified in the brackets.

Table 1 demonstrates that MAGS overcomes the DIRECT and DIRECTl methods on all classes of problems in terms of the averaged number of iterations. In the

[3] In the case of MAGS, the stop condition was different slightly — the global minimizer y^* was considered to be found, if MAGS generates a trial point y^k in the δ-vicinity of the global minimum, i. e. $||y^k - y^*|| \leq \delta$. The size of the vicinity was selected as $\delta = \|b - a\| \sqrt[N]{\Delta}$.

TABLE 2. Averaged numbers of iterations executed by GPMAGS for solving the test optimization problems

		p	$N=4$		$N=5$	
			Simple	*Hard*	*Simple*	*Hard*
I	**Seriquential trial computations**	1	11953	25263	15920	>148342 (4)
II	**Parallel computations** on CPU	2	4762	11178	13378	109075
		4	2372	5972	5203	51868
		8	1393	2874	3773	51868
III	**Parallel computations** on Xeon Phi	60	171	393	382	3452
		120	85	182	249	1306
		240	42	103	97	381

5-Hard class, all three methods failed to solve some problems: DIRECT did not solve 16 problems, DIRECTl and MAGS — 4 problems each.

5.2. **Parallel Computations Using Intel Xeon Phi processors.** As in the previous case, the numerical experiments were performed by using the Simple and Hard function classes with the dimensions equal to 4 and 5. The values of the parameter $\Delta = 10^{-6}$ at $N = 5$ and $\Delta = 10^{-7}$ at $N = 5$ were used. When using the GPMAGS method, the values of the reliability parameter $r = 4.5$ was selected for the Simple class and $r = 5.6$ was selected for the Hard class.

The results of the numerical experiments with GPMAGS with Intel Xeon Phi processors are presented in Tables 5.2 and 5.2.

In the first series of experiments, the serial computations using MAGS were executed. The average numbers of iterations performed by the method for solving the series of optimization problems for each class are shown in row I.

In the second series of experiments, the parallel computations were executed using the CPUs. The achieved relative speedup in iterations is shown in row II; the speedup of the parallel computations is shown relative to the serial computations ($p = 1$).

The final series of experiments was executed using Xeon Phi processors. The results of these experiments are shown in row III. In this case, the speedup factors were calculated relative to the results of the PMAGS method executed on the CPU using eight cores ($p = 8$).

5.3. **Parallel Computations Using General Purpose Graphic Processors.** In the first series of experiments, single graphic accelerator was used for solving 100 6-dimensional optimization problems. The reliability parameter $r = 4.5$ was selected.

The averaged execution time using GPU was 10.78 sec whereas the averaged execution time using all 16 CPU cores of the computational node was 53.8 sec; almost 5-fold improvement was observed.

A larger scale computational experiment was carried out for the global optimization problems with the dimensions of $N = 8$ and $N = 10$: total 12 nodes with 36 graphic accelerators (three accelerators per the computational node) were employed (total 96,768 CUDA-cores were used).

TABLE 3. Speedup of parallel computations executed by GMAGS

		p	$N=4$		$N=5$	
			Simple	*Hard*	*Simple*	*Hard*
I	**Serquential trial computations.** *Average number of iterations.*	1	11953	25263	15920	>148342 (4)
II	**Parallel computations** of CPU. *Speedup*	2	2.52	2.32	1.21	1.41
		4	5.05	4.24	3.13	2.92
		8	8.68	8.88	4.24	6.66
III	**Parallel computations** of Xeon Phi. *Speedup*	60	8.18	7.37	9.99	6.66
		120	16.316	15.815	15.215	17.317
		240	33.133	27.827	38.838	59.359

The averaged execution time using the GPUs to solve the 8-dimensional problem was 405,6 sec that was 5.9 times less than the execution time for CPU. The averaged time of solving the 10-dimensional problem was 2,055.8 sec. (in this case, the execution time for CPU was not calculated because of the high complexity level of the required computations).

6. **Conclusions.** In this paper, the Globalizer global optimization system for solving the time-consuming global optimization problems using a wide spectrum of high-performance computational systems (the systems with shared and distributed memory, the systems with NVIDIA graphic processors and Intel Xeon Phi coprocessors) was considered. The highly parallel computations were facilitated using various distinctive computational schemes: processing several optimization iterations simultaneously, reducing multidimensional optimization problems using multiple Peano space-filling curves, and the multi-stage computing based on nested block reduction schemes. The numerical experiments have confirmed the efficiency of the Globalizer system.

In future, it is planned to extend the Globalizer system for solving the multicriterial multiextremal multidimensional optimization problems with the nonlinear constraints. As a valuable part of further work, Globalizer would be used for solving the applied optimization problems from various application fields.

REFERENCES

[1] K. A. Barkalov and V. P. 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(1)** (2016), 3–20.

[3] K. Barkalov, V. Gergel and I. Lebedev, Use of XeonPhi coprocessor for solving global optimization problems, *LNCS*, **9251** (2015), 307–318.

[4] K. Barkalov, V. Gergel, I. Lebedev, A. Sysoev, Solving the global optimization problems on heterogeneous cluster systems, in *CEUR Workshop Proceedings*, **1482** (2015), 411–419.

[5] K. Barkalov, A. Polov inkin, I. Meyerov, S. Sidorov, N. Zolotykh, SVM regression parameters optimization using parallel global search algorithm, *LNCS*, **7979** (2013), 154–166.

[6] 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] R. Ciegis, D. Henty, B. Kågström and J. Zilinskas, *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 and P. Pugliese and Ya. D. Sergeyev, A global optimization technique for checking parametric robustness, *Automatica*, **35** (1999), 1605–1611.
[11] G. Fasano and J. D. Pinter, *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(1)** (2001), 27–37.
[15] M. Gaviano and D.E. Kvasov and D. Lera and Ya.D. Sergeev, Software for generation of classes of test functions with known local and global minima for global optimization, *ACM Trans. Math. Software*, **29(4)** (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. P. Gergel, A software system for multi-extremal optimization, *Eur. J. Oper. Res.*, **65(3)** (1993), 305–313.
[18] V. P. Gergel, A method for using derivatives in the minimization of multiextremum functions, *Comput. Math. Math. Phys.*, **36(6)** (1996), 729–742.
[19] V. P. Gergel, A global optimization algorithm for multivariate functions with lipschitzian first derivatives, *J. Glob. Optim.*, **10(3)** (1997), 257–281.
[20] V. P. Gergel, et al., High performance computing in biomedical applications, *Procedia Computer Science*, **18** (2013), 10–19.
[21] V. P. Gergel, et al., Recognition of surface defects of cold-rolling sheets based on method of localities, *International Review of Automatic Control* , **8(1)** (2015), 51–55.
[22] V. P. Gergel and S. V. 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. Holmström and M. M. Edvall, The TOMLAB optimization environment, *Modeling Languages in Mathematical Optimization*, 369–376, Springer (2004).
[25] R. Horst and H. Tuy, *Global optimization: deterministic approaches*, Springer-Verlag, Berlin (1990).
[26] D. R. Jones, C. D. Perttunen and B. E. Stuckman, Lipschitzian optimization without the Lipschitz constant, *J. Optim. Theory Appl.*, **79(1)** (1993), 157–181.
[27] R. B. Kearfott, GlobSol user guide, *Optim. Methods Softw.*, **24** (2009), 687–708.
[28] D. E. Kvasov and Ya. 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, Ya. 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(7)** (2008), 1217–1229.
[30] D. E. Kvasov, C. Pizzuti and Ya. D. Sergeyev, Local tuning and partition strategies for diagonal go methods, *Numerische Mathematik*, **94(1)** (2003), 93–106.
[31] L. Liberti, Writing global optimization software, in *Nonconvex Optimization and Its Applications* , **84**, 211–262, Springer (2006).
[32] Y. Lin and L. Schrage, The global solver in the LINDO API, *Optim. Methods Softw.*, **24(4-5)** (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(6)** (2014).
[37] M. P. Pardalos, A. A. Zhigljavsky and J. Zilinskas, *Advances in stochastic and deterministic global optimization*, Springer (2016).
[38] J. D. Pinter, *Global optimization in action (Continuous and Lipschitz optimization: algorithms, implementations and applications)*, Kluwer Academic Publishers, Dordrecht (1996).
[39] J. D. Pinter, 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(4)** (1995), 858–870.
[43] Y. D. Sergeyev, Multidimensional global optimization using the first derivatives, *Comput. Math. Math. Phys.*, **39(5)** (1999), 743–752,.
[44] 44. Ya.D. Sergeyev, D.E. Kvasov, Global search based on efficient diagonal partitions and a set of Lipschitz constants, *SIAM Journal on Optimization*, **16(3)** (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(2)** (2001), 123–145.
[46] Y. D. Sergeyev, R. G. Strongin and D. Lera, *Introduction to global optimization exploiting space-filling curves*, Springer (2013).
[47] Ya. D. Sergeyev, D. Famularo and P. Pugliese, Index branch-and-bound algorithm for lipschitz univariate global optimization with multiextremal constraints, *J. Glob. Optim.*, **21(3)** (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(4)** (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] R.G. Strongin and Ya.D. Sergeyev, *Global optimization with non-convex constraints. Sequential and parallel algorithms*, Kluwer Academic Publishers, Dordrecht (2000).
[52] A. Törn and A. Zilinskas, *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).

Received xxxx 20xx; revised xxxx 20xx.

E-mail address: victor.gergel@itmm.unn.ru
E-mail address: konstantin.barkalov@itmm.unn.ru
E-mail address: alexander.sysoyev@itmm.unn.ru