Parallel Dimensionality Reduction for Multiextremal Optimization Problems

Victor Gergel, Vladimir Grishagin(✉), and Ruslan Israfilov

Lobachevsky State University, Gagarin Avenue 23, 603950 Nizhni Novgorod, Russia
gergel@unn.ru, vagris@unn.ru, ruslan@israfilov.com

Abstract. The paper is devoted to consideration of numerical global optimization methods in the framework of the approach of reducing dimensionality based on nested optimization schemes. For the adaptive nested scheme being more efficient in comparison with its classical prototype a new algorithm of parallel implementation is proposed. General descriptions of the parallel techniques both for synchronous and asynchronous versions are given. Results of numerical experiments on a set of complicated multiextremal test problems of high dimension are presented. These results demonstrate essential acceleration of asynchronous parallel algorithm in comparison with the sequential version.

Keywords: Multiextremal optimization · Global optimum · Dimensionality reduction · Parallel algorithms

1 Introduction

Global optimization problems aimed at finding the global optimum of multiextremal functions are complicated decision making models and describe many important applications in engineering, economy, scientific researches, etc. (see some examples in [3,9,13,26,30,32,36,43]). The complexity of these problems depends crucially on the dimension (number of model parameters) because in general case the growth of the computational costs measured, for example, in number of objective function evaluations is exponential when increasing the dimension. There exist several approaches to analyzing global optimization problems oriented at different classes of multiextremal functions defined by their specific properties. The wide spectrum of directions in the field of global optimization can be found in the fundamental monographs [23,31,32,35,39,44].

Among the approaches generating efficient algorithms to solving multiextremal optimization problems with objective functions satisfying the Lipschitz condition one can mention the approach based on different partition schemes (component approach) and the class of methods which apply the ideas of reducing multidimensional problems to one or a family of univariate subproblems for solving those by means of well-developed one-dimensional optimization algorithms.

V. Malyshkin (Ed.): PaCT 2019, LNCS 11657, pp. 166–178, 2019.
https://doi.org/10.1007/978-3-030-25636-4_13

In the framework of the component approach the search region is partitioned into several subregions (components), every component are evaluated numerically for the purpose of its efficiency for search continuation, and after that a new iteration is carried out in the most "perspective' subregion. The first class of component methods called characteristical ones was proposed and theoretically investigated in the work [18], and later it was generalized to multidimensional case by many researchers (see, for example, publications [24,25,27,31,32,35]).

As for the approach transforming a multidimensional problem to the univariate case, it includes two different schemes. The first one is based on applying the Peano space-filling curves which are continuous mappings of a multidimensional hypercube onto the unit interval of the real axis [4,14,22,28,29,34,39]. The second scheme reduces a multidimensional problem to a family of univariate subproblems connected recursively (nested optimization) [5,6,11,12,16,37–39]. These schemes can be combined when inside the recursive procedure the subproblems of the less dimensionality are considered and solved by means of Peano mappings [40]. As it has been shown in [19], among algorithms of this type the adaptive scheme of nested optimization has demonstrated the best efficiency.

A promising way to overcome the complexity of the multiextremal optimization problems consists in parallelizing sequential schemes of optimization algorithms. Following this idea, some optimization methods have been proposed (see [2,8,15,17,20,33,39,40]). In this paradigm the usual way consists in performing parallel trials (computations of objective function values) [8,17,20,21,39,40]. The algorithm [2] using multiple Peano mappings performs parallel computations of trial couples corresponding to several Peano evolvents. Very interesting approach is used in parallel branch and bound algorithms which build a hierarchical structure of feasible domain partitions and parallelize the procedure of partitioning. For example, the paper [21] describes a model using threads within one computational node and the publication [1] suggests a parallel strategy of partitioning in distributed memory.

As opposed to above approaches the methods on the base of nested optimization scheme [15,33] implement parallelization by means of parallel performance of internal subtasks. In this paper we consider a parallel algorithm being a generalization of the adaptive scheme of global optimization [11] which belongs to the type of recursive reduction techniques and applies for solving the nested univariate subproblems the information characteristical method [33,39]. The main goal of the work is to describe a new model of parallel computations inside the adaptive scheme realizing "parallelization by subtask" approach and to estimate the effectiveness of parallelizing measured as speedup of the parallel adaptive scheme compared to the sequential one.

The rest of the paper is organized as follows. Section 2 contains the statement of multiextremal optimization problem to be studied and the general algorithm of the nested optimization scheme. Section 3 describes the model of parallelism organization in the framework of the nested adaptive dimensionality reduction. Section 4 presents results of numerical experiments and speedup estimations of the parallel adaptive scheme. The last section concludes the paper.

2 Nested Optimization Scheme

The statement of the optimization problem to be considered is as follows. It is necessary to find in a hyperparallelepiped H of the N-dimensional Euclidean space $\mathbb{R}^N$ the least value (global minimum) F_* of an objective function $F(u)$ and the coordinate $u_* \in H$ of the global minimum (global minimizer). This problem can be written in a symbolical form as

$$F(u) \to \min,\ u = (u_1, \ldots, u_N) \in H \subseteq \mathbb{R}^N, \tag{1}$$

$$H = \{u \in \mathbb{R}^N : a_i \le u_i \le b_i, 1 \le i \le N\}, \tag{2}$$

The objective function $F(u)$ is supposed to satisfy in the search domain H the Lipschitz condition

$$|F(u') - F(u'')| \le L\|u' - u''\|,\ u', u'' \in H, \tag{3}$$

where $L > 0$ is a finite value called Lipschitz constant and $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^N$. Under condition (3) the problem (1)–(2) is, in general case, multiextremal and non-smooth.

The nested scheme of dimensionality reduction served as the source for different global optimization methods [5,6,11,12,16,37–39]. It is based on the known relation [5,39]

$$\min_{u \in H} F(y) = \min_{u_1 \in H_1} \min_{u_2 \in H_2} \cdots \min_{u_N \in H_N} F(u_1, \ldots, u_N), \tag{4}$$

where H_i is a line segment $[a_i, b_i]$, $1 \le i \le N$.

Let us give the general description of the nested scheme introducing recursively a family of reduced function $F^i(\tau_i)$, $\tau_i = (u_1, \ldots, u_i)$, $1 \le i \le N$, in the following manner.

$$F^N(\tau_N) \equiv F^N(u) \equiv F(u), \tag{5}$$

$$F^{i-1}(\tau_{i-1}) = \min_{u_i \in H_i} F^i(\tau_i),\ 2 \le i \le N. \tag{6}$$

Then, instead of minimizing in (1) the N-dimensional function $F(u)$ we can search for the global minimum of the univariate function $F^1(u_1)$ as, in accordance with (4),

$$F_* = \min_{u_1 \in H_1} F^1(u_1). \tag{7}$$

However, any numerical optimization method in the course of solving the problem (7) has to calculate values of the function $F^1(t_1)$. But such a computation at a point $\tilde{t}_1$ requires solving the problem

$$F^2(\tilde{t}_1, t_2) \to \min,\ t_2 \in H_2, \tag{8}$$

which are one-dimensional again as the argument $\tilde{t}_1$ is fixed, and so on. Following this way, we reach the level N, where the problem

$$F^N(\tilde{\tau}_{N-1}, t_N) \to \min,\ t_N \in H_N, \tag{9}$$

is one-dimensional as well because the vector $\tilde{\tau}_{N-1} = (\tilde{t}_1, \ldots, \tilde{t}_{N-1})$ is fixed (its coordinates are given at previous levels of recursion). As $F^N(t) \equiv F(t)$ then evaluation of objective function values in the problem (9) consists in calculation of the values $F(\tilde{\tau}_{N-1}, t_N)$ of the given function from (1).

The procedure (7)–(9) described above is recursive and enables to find the solution of the multidimensional problem (1)–(2) via solving the family

$$F^i(\tau_{i-1}, u_i) \to \min,\ u_i \in H_i, 1 \le i \le N, \tag{10}$$

of univariate subproblems. Such the scheme is called *the nested scheme of dimensionality reduction*.

The recursive structure of generation of the subproblems in the family (10) can be presented as a tree of connections between generating (parental) and generated (child) subtasks (see Fig. 1).

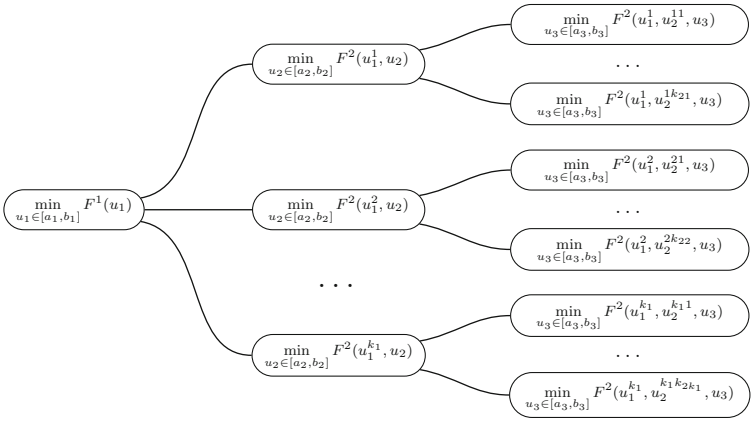

Fig. 1. Tree of subtasks in the nested optimization scheme for dimension 3.

In this tree the problem (7) is the root one and the problems (9) are leaves of the tree. Of course, the tree is built in dynamics, and Fig. 1 shows the full tree obtained after completing multidimensional optimization. It should be noted that conducting one trial (computation of objective function value at a point) in one-dimensional subproblem of minimization of $F^i(\tau_{i-1}, u_i)$, $1 \le i \le N-1$, generates a subtree in the tree of subtasks. As a consequence, any subproblem (10) is parental for subproblems in subtrees generated by its trials.

In classical implementation of the nested scheme the subproblems (10) are solved until a stopping rule of applied univariate method holds true for all of them. It means that in the course of optimization only subproblems which belong to a sole path from the root to a leaf can interact inter se. It leads to loss of search information obtained in the course of optimization and worsens the efficiency of classical scheme significantly.

In order to overcome this drawback of the classical nested scheme, its improved version called *adaptive nested scheme of dimensionality reduction* has been proposed in the paper [11]. As opposed to the classical nested scheme, the adaptive extension considers all the currently existing subproblems (10) simultaneously. A numerical value of "significance" called characteristic is assigned to each subproblem of the current family and all the subproblems are decreasingly ordered according to their characteristics. Then, the subproblem with the maximal characteristic is chosen, and in the best subproblem a new iteration of the univariate method connected with this subproblem is executed. The detailed algorithm of the sequential adaptive scheme has been described in [11].

3 Parallel Adaptive Scheme

A natural way of parallelizing the adaptive scheme consists in solving several subproblems in parallel. Let us suppose that at our disposal there are $P > 1$ parallel computational nodes (processors). Then a parallel iteration of the adaptive scheme could be organized as follows. All the subproblems (subtasks) are ordered according to their characteristics, P subproblems with maximal characteristics are chosen, are distributed among processors (one subproblem to one computational node) and are solved in parallel. Solving within parallel iteration one subproblem of a recursion level l means the decision rule implementation of univariate optimization algorithm used in this subproblem, i.e., the choice of a point u_l of new trial and computation of the objective function value at this point. If $l < N$, such the computation generates a subtree of new subtasks that will be added to existing ones after completing the trial at the point u_l. Hereinafter the operation of executing a trial in a subproblem will be denoted as ExecuteIteration.

If the next parallel iteration will start after completing the work of all processors this procedure corresponds to the synchronous case. However, in such organization of parallelism a processor completing computations is obliged to wait until the other processors finish and will stand idle. To avoid this drawback one can to use more effective, but more complicated asynchronous organization of parallelism when a processor completing its work take the best subtask from the pool of non- distributed subproblems.

Further we consider more detailed how both synchronous and asynchronous parallelisms can be organized for the nested adaptive scheme. As a detailed code of the parallel implementation is very large we will give a general algorithmic description of parallel adaptive scheme on the base of an abstract one-dimensional optimization method. For this formal description it is necessary to introduce several notions and designations.

Let at a stage of the adaptive scheme implementation all subproblems renumber with integer numbers from 1 to λ, where λ is the number of subtasks (10) generated already and the root subproblem (7) is the first one. A univariate method in the course of minimizing a subproblem $F^l(\tau_{l-1}, u_l)$ generates a sequence of trial points $u_l^1, \ldots, u_l^k$ at which the values $z_l^1, \ldots, z_l^k$ are computed where $z_l^i = F^l(\tau_{l-1}, u_l^i), 1 \le i \le k$. These points and values form the set of pairs

$$\omega_k = \{(u_l^1, z_l^1), \ldots, (u_l^k, z_l^k)\}, \tag{11}$$

that can be interpreted as the current state of the search for this subproblem. It should be remembered that any computation of value z_l^i requires solving a one-dimensional subproblem at the next $(l+1)$-th level and, if $l < N$, building a subtree of subproblems (10). Uniting the subtrees of all trials we get the subtree generated by the subproblem on the whole.

Taking these circumstances into account, we identify a subtask $t \in \{1, \ldots, \lambda\}$ as a tuple

$$t = \langle l, \tau_{l-1}, k, \omega_k, h, t^p, T^c, W \rangle. \tag{12}$$

Here l is the number of the recursion level, which the subproblem belongs to, τ_{l-1} is a vector of fixed coordinates obtained from preceding levels, k is the number of trials executed by the univariate algorithm, ω_k from (11). The indicator $h = h(\omega_k)$ shows whether solving this subproblem has been completed, namely, if $h = 0$ then the algorithm solving the described subproblem terminates its execution, if $h = 1$ the optimization has to be continued. The number t^p corresponds to the parental subproblem having generated the current one, and, finally, T^c presents the set of all subtasks (up to the level N) generated by the subproblem considered and, finally, W is a numerical characteristic of the subproblem significance. The set of all subtasks t, $t \in \{1, \ldots, \lambda\}$, we will denote as T.

It should be noted that tuple (12) is not applicable to the root subtask (7) because it has no parents. In order to include the root subproblem into the unified description let us introduce as a parent of the root an "empty" subtask t^0 and define $t_0 = \varnothing$.

For starting the adaptive scheme (both sequential and parallel) it is necessary to create an initial set T. It could be done applying the classical nested scheme with a few trials in one dimensional search. We will consider just a general procedure INITIALIZE implementing this initial stage without its concretization. It is executed only once and it is not important whether it is sequential or parallel.

As for parallel implementation of the main body of the adaptive scheme we will deal with a computational system with distributed memory. The system is supposed to consist of P computational nodes. Each node has just one processor and memory, to which the processor of the node only has the direct access. The remote direct access to this memory (RDMA) is considered to be impossible. It means that recording the data of j-th node in the memory of i-th node $(i \neq j)$ can be carried out by means of operations of data transmission only.

The simplest way of parallelizing the adaptive scheme in a distributed system can consist in employment of the program model MapReduce [7]. A generalized algorithm of the parallel adaptive scheme could be presented as the Algorithm 1.

Algorithm 1. Parallel adaptive scheme on the base of MapReduce

1: $l \leftarrow 1,\ t^1 \leftarrow 1$
2: $T \leftarrow$ INITIALIZE()
3: **while** $h(t^1) = 1$ **do**
4: $\quad T' \leftarrow \{ t_1, \ldots, t_P : W(t) \leq W(t_i),\ 1 \leq i \leq P,\ t \in T \setminus \{t_1, \ldots, t_P\} \}$
5: $\quad T'' \leftarrow$ MAPREDUCE(T', EXECUTEITERATION)
6: $\quad T \leftarrow T \cup T''$
7: $\quad T \leftarrow T \setminus \{t \in T' : h(t) = 0\}$
8: **end while**

In the Algorithm 1, after initialization in the loop until the termination condition in the root subproblem is satisfied parallel iterations of the adaptive scheme are executed. At Stage 4 the set T' containing P subtasks with the best characteristics is formed. Stage 5 distributes the subproblems from T' to processors which in parallel execute one trial in their subtasks with the help of procedure EXECUTEITERATION. After completing all the trials a set T'' of new subproblems obtained in the course of computations is formed. Stage 6 complements the set T with new subproblems and Stage 6 removes from the set T the terminated subproblems.

Practical implementation of the described algorithm can be realized in the framework of such the platforms as Hadoop [41] or Spark [42]. Unfortunately, this algorithm is synchronous and requires significant number of data transmissions. Moreover, implementation of Algorithm 1 implies that one processor (master node) plays the main role and coordinates the work of the other (slave) processors, i.e., the organization of the parallel processes is centralized.

To improve the parallel implementation of the adaptive scheme we propose for the adaptive scheme an asynchronous decentralized model of parallel computations where all processors are equal in rights.

Let, as earlier, a distributed system have P processors. We change Algorithm 1 so that procedure INITIALIZE after creating an initial set T splits this set into P parts and send each part to separate processor. Moreover, i-th processor is supposed to be able to connect independently with any other node and to execute the information interchange with it after completing a trial in its subproblems. Under this assumption the full set T of subtasks can be stored portionwise on different nodes and in order to get the best subproblems, a node can request from other nodes only the best subproblems from their local subsets.

In this situation there exists no integrated iteration implemented by all the processors jointly and we can deal with iterations executed by processors separately. Completing its iteration a node can request immediately the best subproblems from other nodes and begin a new iteration. Two examples of requests are shown in Fig. 2.

Under these assumptions we propose an asynchronous algorithm of the parallel adaptive scheme that is presented below in a pseudo code. This algorithm is supposed to be executed on every node.

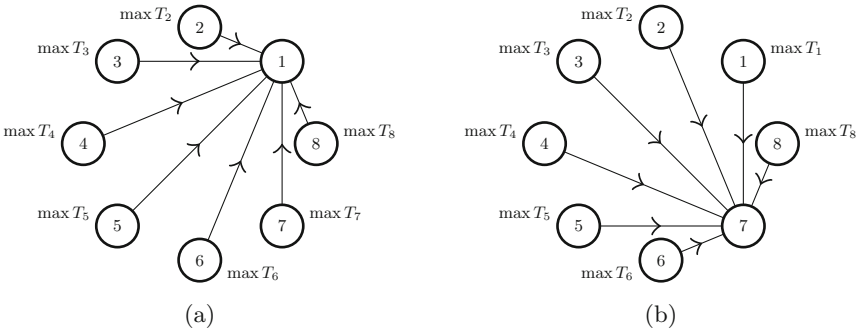

Fig. 2. Gathering information for new iteration to the 1-st node (a) and to the 7-th node (b).

Algorithm 2. Parallel asynchronous adaptive scheme

```
 1: procedure ReceiveTaskFromNode(j)
 2:     t_j^* ← max T_j^local
 3:     T_j^local ← T_j^local \ { t_j^* }
 4:     return t_j^*
 5: end procedure
 6:
 7: procedure RunOnNode(i)
 8:     T_i^local ← PartOfInitialTaskSet()
 9:     for n ∈ {1, 2, ..., n_max^i} do
10:         for j ∈ {1, 2, ..., P} \ { i } do
11:             t_j^* ← ReceiveTaskFromNode(j)
12:             T_i^local ← T_i^local ∪ { t_j^* }
13:         end for
14:         t^* ← max T_i^local
15:         t^1, T^1 ← ExecuteIteration(t^*)
16:         T_i^local ← T_i^local ∪ T^1
17:         if h(t^*) = 0 then
18:             if l(t^*) = 1 then
19:                 BroadcastStopSignal()
20:                 break
21:             end if
22:             T_i^local ← T_i^local \ { t^* }
23:         end if
24:     end for
25: end procedure
```

Let us give some remarks about the Algorithm 2. T_i^{local} is the subset of subproblems stored on the i-th node. Procedure ReceiveTaskFromNode provides receiving the best subtask from other node. In line 8 the procedure PartOfInitialTaskSet forms the initial set T_i^{local} from subtasks obtained by the procedure Initialize.

The external loop **for** is an analogue of the loop **while** in Algorithm 1 but instead of termination condition used in **while** a limit $n_i^{\max}$ of trials executed on the i-th node is introduced. Termination condition of the optimization algorithm is transferred into lines 17–21, where $l(t)$ denotes the level of the subtask t. The internal loop carries out collecting the subproblems with the best characteristics from all the nodes (except for the i-th node). Further, from the local set T_i^{local} the subproblem with the maximal characteristic is chosen and the new trial in this subproblem is executed.

If during solving a problem (1)–(2) i-th processor has performed n_i trials, it has transferred tasks to other processors $(P-1)n_i$ times because one trial requires $P-1$ transmissions from the rest of nodes. Altogether the processors have performed $n = \sum_{i=1}^{P} n_i$ trials and, consequently, executed $(P-1)n$ transmissions. The estimation of transmissions number for synchronous Algorithm 1 gives the result about $(P^2+P)n$, which is worse essentially compared with the asynchronous case.

4 Numerical Experiments

To evaluate the effectiveness of the parallelism described in Sect. 3 a computational experiment aimed at comparison of sequential and asynchronous parallel implementations of the adaptive nested optimization schemes has been carried out. The simpler synchronous version did not participated in comparison because it is inferior to the asynchronous one according to theoretical estimations. The experiment consisted in solving a set of functions from the test class GKLS [10] of essentially multiextremal functions (hard subclass). These functions have complicated structure with tens of local minima. Nowadays this class is a classical tool for comparison of global optimization methods.

In experiment 50 functions of dimension 8 have been taken and solved both sequential and parallel adaptive schemes. As one-dimensional method for solving the subproblems 10 in both the schemes the information global search algorithm GSA [38,39] was taken with reliability parameter $r = 6.5$ and accuracy in termination condition $\varepsilon = 0.01$. Computations were executed on the cluster consisting of 64 nodes, where each node is equipped with Intel® Xeon® Gold 6148 processor having 20 physical cores. Mellanox® Infiniband FDR was used as interconnection technology. The parallelism was provided on the base of MPI, version Intel® MPI 2019. Only one MPI rank was assigned to one node.

The global minima have been found with given accuracy in all the test problems. The results of the experiment are reflected in Table 1 and Fig. 3. The table contains average time spent by the parallel scheme per one test problem and speedup in time achieved by the parallel technique in comparison with the sequential one for different number of MPI ranks.

Table 1. Speedup in time on GKLS test class

MPI rank	1	2	4	8	16	32	64
Time (sec.)	7409.39	4919.94	2422.97	1202.00	717.19	380.39	198.88
Speedup	1	1.50	3.05	6.16	10.33	19.47	37.26

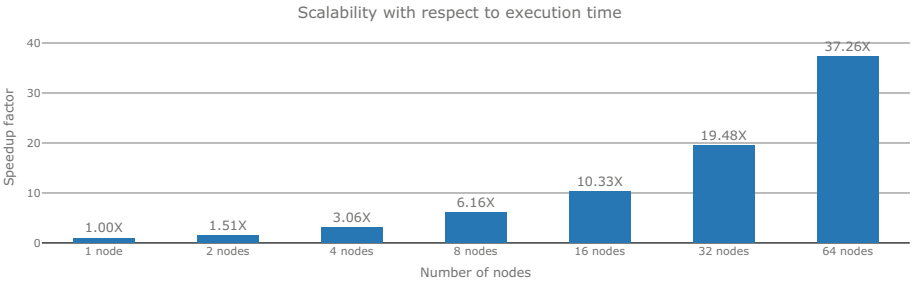

Fig. 3. Speedup in time on GKLS test class

5 Conclusion

The paper proposes general descriptions of new parallel algorithms implementing methods of multiextremal optimization on the base of adaptive nested schemes reducing a multidimensional problem to a family of one-dimensional subproblems. Two parallel versions are presented in synchronous and asynchronous variants for computational distributed systems. Efficiency of the parallelism are investigated experimentally on the test class GKLS of complicated multiextremal multidimensional problems. The results of the experiment have shown essential speedup of the optimization process in case of applying the asynchronous adaptive scheme.

Combining the general parallel procedure of the adaptive scheme with fast univariate optimization methods (like characteristical ones) enables to construct new efficient techniques for solving multiextremal problems of high dimensions. Moreover, it is promising to develop new parallel implementations of the adaptive scheme oriented at other parallel architectures, for example, at supercomputers with mixed types of memory. It would be very interesting as well to compare the proposed algorithm with parallel optimization methods based on the other principles of parallelizing. These problems can be fruitful directions of further researches.

Acknowledgements. The research has been supported by the Russian Science Foundation, project No 16-11-10150 “Novel efficient methods and software tools for time-consuming decision make problems using superior-performance supercomputers”.

References

1. Androulakis, I.P., Floudas, C.A.: Distributed branch and bound algorithms for global optimization. In: Pardalos, P.M. (ed.) Parallel Processing of Discrete Problems. The IMA Volumes in Mathematics and its Applications, vol. 106, pp. 1–35. Springer, New York (1999). https://doi.org/10.1007/978-1-4612-1492-2_1
2. Barkalov, K., Gergel, V.: Parallel global optimization on GPU. J. Glob. Optim. **66**(1), 3–20 (2016). https://doi.org/10.1007/s10898-016-0411-y
3. Bartholomew-Biggs, M., Parkhurst, S., Wilson, S.: Using direct to solve anaircraft routing problem. Comput. Optim. Appl. **21**(3), 311–323 (2002). https://doi.org/10.1023/A:1013729320435
4. Butz, A.R.: Space-filling curves and mathematical programming. Inform. Control **12**, 314–330 (1968)
5. Carr, C.R., Howe, C.W.: Quantitative Decision Procedures in Management and Economic: Deterministic Theory and Applications. McGraw-Hill, New York (1964)
6. Dam, E.R., Husslage, B., Hertog, D.: One-dimensional nested maximin designs. J. Glob. Optim. **46**, 287–306 (2010)
7. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: Sixth Symposium on Operating System Design and Implementation, OSDI 2004, San Francisco, CA, pp. 137–150 (2004)
8. Evtushenko, Y.G., Malkova, V.U., Stanevichyus, A.A.: Parallel globaloptimization of functions of several variables. Comput. Math. Math. Phys. **49**(2), 246–260 (2009). https://doi.org/10.1134/S0965542509020055
9. Famularo, D., Pugliese, P., Sergeyev, Y.: A global optimization technique for checking parametric robustness. Automatica **35**, 1605–1611 (1999)
10. Gaviano, M., Kvasov, D.E., Lera, D., Sergeyev, Y.D.: Software for generation ofclasses of test functions with known local and global minima for globaloptimization. ACM Trans. Math. Softw. **29**(4), 469–480 (2003)
11. Gergel, V.P., Grishagin, V.A., Gergel, A.V.: Adaptive nested optimization scheme for multidimensional global search. J. Glob. Optim. **66**, 35–51 (2016)
12. Gergel, V.P., Grishagin, V.A., Israfilov, R.A.: Local tuning in nested scheme of global optimization. Proc. Comput. Sci. **51**, 865–874 (2015)
13. Gergel, V.P., Kuzmin, M.I., Solovyov, N.A., Grishagin, V.A.: Recognition of surface defects of cold-rolling sheets based on method of localities. Int. Rev. Autom. Control **8**, 51–55 (2015)
14. Goertzel, B.: Global optimization with space-filling curves. Appl. Math. Lett. **12**, 133–135 (1999)
15. Grishagin, V.A., Israfilov, R.A.: Multidimensional constrained global optimization in domains with computable boundaries. In: CEUR Workshop Proceedings, vol. 1513, pp. 75–84 (2015)
16. Grishagin, V.A., Israfilov, R.A.: Global search acceleration in the nested optimization scheme. In: AIP Conference Proceedings, vol. 1738, p. 400010 (2016)
17. Grishagin, V.A., Sergeyev, Y.D., Strongin, R.G.: Parallel characteristical algorithms for solving problems of global optimization. J. Glob. Optim. **10**, 185–206 (1997)
18. Grishagin, V.: On convergence conditions for a class of global search algorithms. In: Proceedings of the 3-rd All-Union Seminar Numerical Methods of Nonlinear Programming, pp. 82–84 (1979, in Russian)

19. Grishagin, V., Israfilov, R., Sergeyev, Y.: Convergence conditions and numerical comparison of global optimization methods based on dimensionality reduction schemes. Appl. Math. Comput. **318**, 270–280 (2018). https://doi.org/10.1016/j.amc.2017.06.036. http://www.sciencedirect.com/science/article/pii/S0096300317304496. Recent Trends in Numerical Computations: Theory and Algorithms
20. He, J., Verstak, A., Watson, L.T., Sosonkina, M.: Design and implementation of a massively parallel version of DIRECT. Comput. Optim. Appl. **40**(2), 217–245 (2008). https://doi.org/10.1007/s10589-007-9092-2
21. Herrera, J.F.R., Salmerón, J.M.G., Hendrix, E.M.T., Asenjo, R., Casado, L.G.: On parallel branch and bound frameworks for global optimization. J. Glob. Optim. **69**(3), 547–560 (2017). https://doi.org/10.1007/s10898-017-0508-y
22. Hime, A., Oliveira Jr., H., Petraglia, A.: Global optimization using space-filling curves and measure-preserving transformations. Soft Comput. Industr. Appl. **96**, 121–130 (2011)
23. Horst, R., Pardalos, P.M.: Handbook of Global Optimization. Kluwer Academic Publishers, Dordrecht (1995)
24. Jones, D.R.: The DIRECT global optimization algorithm. In: Floudas, C., Pardalos, P.M. (eds.) Encyclopedia of Optimization, pp. 431–440. Kluwer Academic Publishers, Dordrecht (2001)
25. Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. J. Optim. Theory Appl. **79**, 157–181 (1993)
26. Kvasov, D.E., Menniti, D., Pinnarelli, A., Sergeyev, Y.D., Sorrentino, N.: Tuning fuzzy power-system stabilizers in multi-machine systems by global optimization algorithms based on efficient domain partitions. Electr. Power Syst. Res. **78**, 1217–1229 (2008)
27. Kvasov, D.E., Pizzuti, C., Sergeyev, Y.D.: Local tuning and partition strategies for diagonal GO methods. Numer. Math. **94**, 93–106 (2003)
28. Lera, D., Sergeyev, Y.D.: Lipschitz and Hölder global optimization using space-filling curves. Appl. Numer. Math. **60**, 115–129 (2010)
29. Lera, D., Sergeyev, Y.D.: Deterministic global optimization using space-filling curves and multiple estimates of Lipschitz and holder constants. Commun. Nonlinear Sci. Numer. Simul. **23**, 328–342 (2015)
30. Modorskii, V.Y., Gaynutdinova, D.F., Gergel, V.P., Barkalov, K.A.: Optimization in design of scientific products for purposes of cavitation problems. In: AIP Conference Proceedings, vol. 1738, p. 400013 (2016)
31. Paulavičius, R., Žilinskas, J.: Simplicial Global Optimization. Springer, NewYork (2014). https://doi.org/10.1007/978-1-4614-9093-7
32. Pintér, J.D.: Global Optimization in Action. Kluwer Academic Publishers, Dordrecht (1996)
33. Sergeyev, Y.D., Grishagin, V.A.: Parallel asynchronous global search and the nested optimization scheme. J. Comput. Anal. Appl. **3**, 123–145 (2001)
34. Sergeyev, Y.D., Strongin, R.G., Lera, D.: Introduction to Global Optimization Exploiting Space-Filling Curves. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-8042-6
35. Sergeyev, Y., Kvasov, D.: Deterministic Global Optimization: An Introduction to the Diagonal Approach. Springer, New York (2017). https://doi.org/10.1007/978-1-4939-7199-2

36. Shevtsov, I.Y., Markine, V.L., Esveld, C.: Optimal design of wheel profile for railway vehicles. In: Proceedings of the 6th International Conference on Contact Mechanics and Wear of Rail/Wheel Systems, Gothenburg, Sweden, pp. 231–236 (2003)
37. Shi, L., Ólafsson, S.: Nested partitions method for global optimization. Oper. Res. **48**, 390–407 (2000)
38. Strongin, R.G.: Numerical Methods in Multiextremal Problems (Information-Statistical Algorithms). Nauka, Moscow (1978, in Russian)
39. Strongin, R.G., Sergeyev, Y.D.: Global Optimization with Non-convex Constraints: Sequential and Parallel Algorithms. Kluwer Academic Publishers/Springer, Dordrecht/Heiselberg (2014)
40. Sysoyev, A., Barkalov, K., Sovrasov, V., Lebedev, I., Gergel, V.: Globalizer – a parallel software system for solving global optimization problems. In: Malyshkin, V. (ed.) PaCT 2017. LNCS, vol. 10421, pp. 492–499. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62932-2_47
41. White, T.: Hadoop: The Definitive Guide. O'Reilly Media, Inc., Newton (2009)
42. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud 2010, p. 10. USENIX Association, Berkeley (2010). http://dl.acm.org/citation.cfm?id=1863103.1863113
43. Zhao, Zh., Meza, J.C., Van Hove, M.: Using pattern search methods for surface structure determination of nanomaterials. J. Phys.: Condens. Matter **18**(39), 8693–8706 (2006)
44. Zhigljavsky, A.A., Žilinskas, A.: Stochastic Global Optimization. Springer, NewYork (2008). https://doi.org/10.1007/978-0-387-74740-8