High performance computing for global optimization problems

K.A. Barkalov, V.P. Gergel

National Research Lobachevsky State University of Nizhny Novgorod, Gagarin avenue 23, Nizhny Novgorod, Russia, 603950

Abstract. In the present work, the multiextremal optimization problems and a high-performance parallel algorithm for solving these ones are considered. The investigation of the algorithm scalability has been carried out on the problem class, in which the computation costs of the functions depended on the iteration point. The algorithm proposed in the present work can utilize the CPUs (for solving more complex subproblems) as well as the GPUs (for solving the simple subproblems). The results of numerical experiments demonstrating the speedup when solving a series of multiextremal constrained problems are presented.

1. Introduction

In the present paper, the constrained multiextremal optimization problems and the parallel methods for solving these ones are considered. The fact that the global extremum is an integral characteristic of a problem is an important feature of the multiextremal problems. Thus, its finding is related to the construction of a coverage of the search domain and the computing the optimized function values at all points of this coverage. The dimensionality affects the difficulty of solving the problems of considered class crucially: the computational costs grow with increasing this one exponentially, therefore, for solving such problems, the methods are required, which generate a nonuniform grid in the search domain, denser near the global minimizer and more sparse far away from this one. In the present paper, the results of scalability investigation of the parallel index global optimization algorithm developed in Lobachevsky State University of Nizhny Novgorod [1, 2] and implemented in the Globalizer system [3] are presented.

Within the framework of the discussed approach, the solving of the multidimensional problems is reduced to the solving of a set of the connected subproblems of less dimensionality. The corresponding dimensionality reduction is based on the use of the evolvents of a unit interval on the real axis onto a hypercube. The continuous unambiguous mappings like Peano curves, called also the space-filling curves, play the role of such evolvents. The nested optimization scheme is one more mechanism used for reducing the dimensionality of the problem being solved. The numerical methods allowing the efficient utilization of such reduction schemes had been developed in details and substantiated in [1, 2].

In the algorithms considered in the present paper the objective function and constraints are assumed to be Lipschitzian that is typical for many other approaches to the development of the parallel optimization algorithms (see, for example, [4, 5, 6, 7]). This assumption is natural for many applied problems since the relative variations of the functions generally cannot exceed a certain threshold determined by the bounded energy of changes occurring in the system under

study. Also, a standard assumption is that the execution of a single trial (the computing of the values of the objective function and constraints at a point in the search domain) is a time-consuming operation since it utilized the results of numerical modeling.

An assumption on different costs of computing the problem functions subject to various components of the vector of parameters is a novel element investigated in the present work. It is assumed that it is possible to select the "difficult" and "easy" parts of the functions connected with each other. An example of a problem of this type has been considered in [8]. Then, the solving of the "difficult" part of the problem (for which the execution of time-consuming algorithms is required to conduct the trials) can be performed on CPU using an efficient index global optimization algorithm. The "easy" part of the problem (the execution of which doesn't require the complex algorithms and can be transferred to an accelerator easily) can be solved on GPU using the uniform grid technique. Obviously, in this case a major portion of the trials will be carried out on GPU, and the minor one will be carried out on CPU. However, because of the difference in the difficulties of the subproblems solved on different units, a speedup of the algorithm as a whole can be expected. The proposed approach has been implemented in Globalizer parallel software system for solving the global optimization problem developed in Lobachevsky State University of Nizhny Novgorod.

2. Problem statement

Let us consider the N-dimensional global optimization problem

$$\varphi(y^*) = \min\{\varphi(y) : y \in D,\ g_i(y) \le 0,\ 1 \le i \le m\} \tag{1}$$

with search domain

$$D = \left\{y \in R^N : a_j \le y_j \le b_j, 1 \le j \le N\right\}.$$

The objective function $\varphi(y)$ (henceforth denoted by $g_{m+1}(y)$) and the left-hand sides $g_i(y)$, $1 \le i \le m$, of the constraints satisfy the Lipschitz conditions with constants L_i, $1 \le i \le m+1$, respectively, and may be multiextremal. It is assumed that the functions $g_i(y)$, $1 \le i \le m+1$, are defined and computable only in the corresponding domains

$$Q_1 = D,\ Q_{i+1} = \{y \in Q_i : g_i(y) \le 0\},\ 1 \le i \le m.$$

These conditions allow for the introduction of a classification of the points $y \in D$ according to the number $\nu(y)$ of the constraints computed at this point.

Thus, a *trial* at a point $y^k \in D$ executed at the k-th iteration of the algorithm will consist in computing the values $g_1(y), ..., g_\nu(y)$, where the index $\nu \le m$ is determined by the conditions

$$g_i(y^k) \le 0,\ 1 \le i < \nu,\ g_\nu(y^k) > 0,\ \nu \le m.$$

The occurrence of the first violation of the constraint terminates the trial. In the case when the point y^k is a feasible one, i.e. when $y \in Q_{m+1}$, the trial includes the computation of the values of all the functions of the problems and the index is assumed to be $\nu = m+1$. The pair of values

$$\nu = \nu(y^k),\ z^k = g_\nu(y^k)$$

is a *trial result*.

The parallel index algorithm, which can be applied to solving such problems with partially defined functions and which has been implemented in the Globalizer system, described in details in [2]. The main idea of the algorithm consists in the reduction of the initial multidimensional problem to a set of the nested subproblems of less dimensionality and the solving of these ones in parallel. The dimensionality reduction schemes utilized in the operation of the index algorithm are described briefly in the next section.

3. Dimensionality reduction

3.1. Dimensionality reduction using space-filling curves

The use of Peano curve $y(x)$

$$\left\{y \in R^N : -2^{-1} \le y_i \le 2^{-1}, 1 \le i \le N\right\} = \{y(x) : 0 \le x \le 1\}$$

unambiguously mapping the interval of real axis $[0,1]$ onto a N-dimensional cube is the first of the dimensionality reduction methods considered. To implement this method of dimensionality reduction a numerically constructed curve (*evolvent*) is used. The evolvent is 2^{-m} accurate approximation of the theoretical Peano curve in L_∞ metric, where m is an evolvent construction parameter. Problems of numerical construction of the evolvents and the corresponding theory are considered in detail in [1].

By using this kind of mapping it is possible to reduce the multidimensional problem (1) to a univariate problem

$$\varphi(y(x^*)) = \min\left\{\varphi(y(x)) : x \in [0,1],\ g_i(y(x)) \le 0,\ 1 \le i \le m\right\}.$$

The considered dimensionality reduction scheme juxtaposes a multidimensional problem with lipschitzian functions to a univariate problem where the corresponding functions satisfy the uniform Hölder condition (see [1]), i.e.

$$\left|g_i(y(x')) - g_i(y(x''))\right| \le H_i \left|x' - x''\right|^{1/N},\ x', x'' \in [0,1],\ 1 \le i \le m+1.$$

Here N is the dimensionality of the initial multidimensional problem and the coefficients H_i are related with the Lipschitz constants L_i of the initial problem by the inequalities $H_i \le 2L_i\sqrt{N+3}$.

3.2. Nested optimization scheme

Nested optimization scheme is based on relation (see [2])

$$\min_{y \in D}\{\varphi(y) :\ g_i(y) \le 0,\ 1 \le i \le m\} = \min_{u_1 \in D_1}\min_{u_2 \in D_2} ... \min_{u_M \in D_M}\{\varphi(y) :\ g_i(y) \le 0,\ 1 \le i \le m\},$$

which allows replacing the solving of multidimensional problem (1) by solving a family of subproblems related to each other recursively. Here we consider vector y as a vector of block variables

$$y = (y_1, y_2, ..., y_N) = (u_1, u_2, ..., u_M),$$

where the i-th block variable u_i is a vector of N_i components of vector y, taken serially, i.e. $u_1 = (y_1, y_2, ..., y_{N_1})$, $u_2 = (y_{N_1+1}, y_{N_1+2}, ..., y_{N_1+N_2})$,..., $u_M = (y_{N-N_M+1}, y_{N-N_M+2}, ..., y_N)$, at that $N_1 + N_2 + ... + N_M = N$. The subdomains $D_i, 1 \le i \le M$, are projections of initial search domain D onto the subspaces corresponding to the variables $u_i, 1 \le i \le M$.

In the present study, this scheme has been applied for $M = 2$, i.e. only one nesting level has been used

$$\min_{y \in D}\{\varphi(y) :\ g_i(y) \le 0,\ 1 \le i \le m\} = \min_{u_1 \in D_1}\min_{u_2 \in D_2}\{\varphi(y) :\ g_i(y) \le 0,\ 1 \le i \le m\}.$$

4. Organization of parallel computing

The organization of parallel computing with the use of the recursive optimization scheme has been considered in details for the shared/distributed memory as well as for the accelerators in [9, 10]. However, in these works the problems were considered, in which the time of the trial execution didn't depend on the trial point. Here, the problem is considered assuming that in the

function $\varphi(y_1, ..., y_N)$ it is possible to select more difficult part $f(y_1, ..., y_s)$ (depending on a part of the parameters only) and a simpler part $g(y_1, ..., y_N)$ (depending on all problem parameters), for example, $\varphi(y) = f(y)g(y)$.

The difficult part of the function implies performing some computations related to the numerical simulation, which can be performed on the CPUs only. The simple part doesn't imply the complex computations and can be computed on an accelerator, for example, GPU. For solving a problem with such a structure, one can apply the parallel recursive optimization scheme with the use CPU at the upper nesting level and GPU at the lower one.

5. Numerical experiments

The recursive scheme of solving the global optimization problems has been implemented in the Globzlizer solver developed in Lobachevsky State University of Nizhny Novgorod [3]. The global search methods and various dimensionality reduction schemes make the algorithmic base for the Globalizer. The numerical experiments, the results of which are presented in [12, 11] demonstrate these methods, at least, are not worse than the well known methods applied for similar purposes, and even overcome these ones with respect to some parameters.

Let us conduct the investigation of the scalability of the parallel algorithm by solving a series of 100 test problems of the constrained global optimization. In work [13] the approach has been proposed, which allowed generating the constrained global optimization problems with the following properties:

- one could control the size of the feasible domain Q_{m+1} with respect to the whole domain D;
- the global minimizer of the objective function is known a priori taking into account the constraints;
- the global minimizer of the objective function without accounting for the constraints is out of the feasible domain Q_{m+1} (with the purpose of simulating the behavior of the constraints and the objective function in the applied constrained optimization problems).

In order to simulate the applied problems with various computational difficulty, we will use a combination of the function classes of the kind

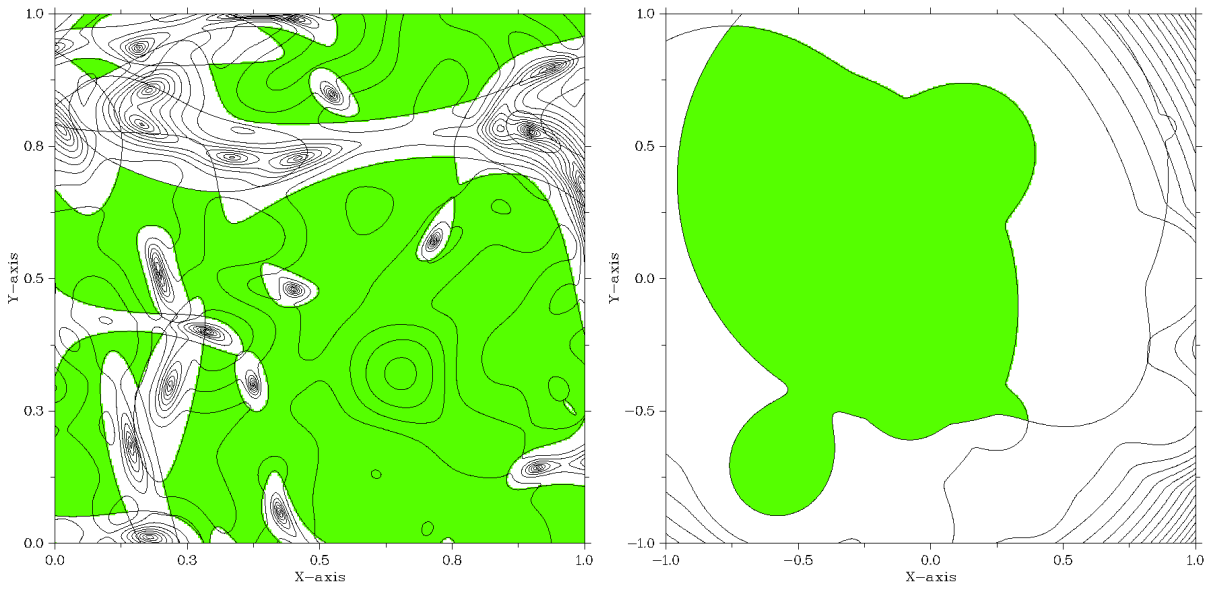
$$\varphi(y_1, ..., y_N) = p(y_1, ..., y_N)(f(y_1, y_2) + g(y_3, ..., y_N)).$$

Here $f(y_1, y_2)$ is a two-dimensional function from the class described in [14], $g(y_3, ..., y_N)$ is a function of the dimensionality $N-2$ from the class described in [15], and $p(y_1, ..., y_N)$ is a second order polynomial. The multiplication by $p(y_1, ..., y_N)$ excluded the possibility of separable search of the minima of the functions $f(y_1, y_2)$ and $g(y_3, ..., y_N)$.

The level lines of the two-dimensional subproblems based on the functions $f(y_1, y_2)$ and $g(y_3, y_4)$ are shown in figure 1. The feasible domains are highlighted by color. One can see that the subproblem (a) has more complex structure as compared to the subproblem (b). At the same time, computing the values of the function $f(y_1, y_2)$ is more time-consuming than of $g(y_3, y_4)$.

The numerical experiments were carried out using two classes of problems (*Simple* ans *Hard*) with $N = 5$. The problem was considered to be solved if the algorithm generates a trial point y^k in δ-vicinity of the global minimizer, i.e. $\|y^k - y^*\| \leq \delta$. The size of the vicinity was selected as $\delta = 0.01 \|b - a\|$, where a and b are the boundaries of the search domain D. The maximum allowable number of iterations was $K_{max} = 10^6$.

Let us conduct the first experiment by solving the *Simple* and *Hard* problem series on a single node employing both available CPUs in full (i. e. $p = 16$ cores available). In table 1 the averaged time (in seconds) required to solve the problems of the series is presented.



(a) (b)

Figure 1. The level lines of (a) hard and (b) simple subproblems

Table 1. Average time for solving the problem on one node

Problems	Time (sec)
Hard	52
Simple	51

Let us conduct the next experiment employing p nodes each including three graphic accelerators. Thus, at $p = 32$ total 96 GPU accelerators were employed each including 2688 CUDA cores. In tables 2 and 3 the average time and the speedup with respect to the run on a single node are presented.

Table 2. Average time for solving the problem on p nodes

Problems	$p = 8$	$p = 16$	$p = 32$
Hard	11.51	5.53	0.54
Simple	2.04	1.50	0.47

Table 3. Speedup with respect to one node

Problems	$p = 8$	$p = 16$	$p = 32$
Hard	5	9	96
Simple	25	34	109

Computational experiments were carried out on a high-performance cluster of Lobachevsky State University of Nizhny Novgorod. The cluster node included two Intel Sandy Bridge E5-2660 2.2 GHz CPUs, 64 Gb RAM, and three NVIDIA Kepler K20 GPUs (2688 CUDA cores, 6 Gb GDDR5). The results of experiments demonstrate that the parallel index algorithm combined with the nested optimization scheme provides a good speedup on the considered problem class.

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

References

[1] Strongin, R.G. Global optimization with non-convex constraints. Sequential and parallel algorithms. / R.G. Strongin, Ya.D. Sergeyev — Dordrecht: Kluwer Academic Publishers, 2000. — 704 p.
[2] Strongin, R.G. Parallel computing in global optimization problems / R.G. Strongin, V.P. Gergel, V.A. Grishagin, K.A. Barkalov — Moscow: Publishing of the Moscow State University, 2013. — 280 p. — (in Russian).
[3] Sysoyev, A.V. Globalizer – A parallel software system for solving global optimization problems / A.V. Sysoyev, K.A. Barkalov, V.V. Sovrasov, I.G. Lebedev, V.P. Gergel // Lecture Notes in Computer Science. — 2017. — Vol. 10421. — P. 492–499.
[4] Jones, D.R. The DIRECT global optimization algorithm / In: Floudas, C. A., Pardalos, P. M. (eds.) The Encyclopedia of Optimization, Second Edition. — Heidelberg: Springer, 2009. — P. 725–735.
[5] Paulavičius, R. Parallel branch and bound for global optimization with combination of Lipschitz bounds / R. Paulavičius, J. Žilinskas, A. Grothey // Optimization Methods & Software. — 2011. — Vol. 26(3). — P. 487–498.
[6] Evtushenko, Yu.G. Parallel global optimization of functions of several variables / Yu.G. Evtushenko, V.U. Malkova, A.A. Stanevichyus // Computational Mathematics and Mathematical Physics. — 2009. — Vol. 49 (2). — P. 246–260.
[7] Evtushenko, Yu.G. Method of non-uniform coverages to solve the multicriteria optimization problems with guaranteed accuracy / Yu.G. Evtushenko, M. A. Posypkin // Automation and Remote Control. — 2014. — Vol. 75(6). — P. 1025–1040.
[8] Barkalov, K.A. Test problems for parallel algorithms of constrained global optimization / K.A. Barkalov, R.G. Strongin // Lecture Notes in Computer Science. — 2017. — Vol. 10556. — P. 18–33.
[9] Sysoyev, A.V. MPI implementation of dimension reduction multilevel scheme for parallel solving the global optimization problems / A.V. Sysoyev, K.A. Barkalov, V.P. Gergel, I.G. Lebedev // Russian Supercomputing Days: Proceedings of the International Scientific Conference. — Moscow: Publishing of the Moscow State University, 2015. — P. 61–68.
[10] Barkalov, K.A. Solving global optimization problems on GPU / K.A. Barkalov, I.G. Lebedev // Russian Supercomputing Days: Proceedings of the international conference. — Moscow: Publishing of the Moscow State University, 2016. — P. 640–650.
[11] Barkalov, K. Parallel global optimization on GPU / K. Barkalov K., V. Gergel // Journal of Global Optimization. — 2016. — Vol. 66(1). — P. 3–20.
[12] Barkalov, K. Use of Xeon Phi Coprocessor for Solving Global Optimization Problems / K. Barkalov, V. Gergel, I. Lebedev // Lecture Notes in Computer Science. — 2015. — Vol. 9251. — P. 307–318.
[13] Gergel, V. An Approach for Generating Test Problems of Constrained Global Optimization / V. Gergel // Lecture Notes in Computer Science. — 2017. — Vol. 10556. — P. 314–319.
[14] Gergel, V. Adaptive nested optimization scheme for multidimensional global search / V. Gergel, V. Grishagin, A. Gergel // Journal of Global Optimization. — 2016. — Vol. 66(1). — P. 35–51.
[15] Sergeyev, Y.D. (2015) A deterministic global optimization using smooth diagonal auxiliary functions / Y.D. Sergeyev, D.E. Kvasov // Communications in Nonlinear Science and Numerical Simulation. — 2015. — Vol. 21 (1-3). — P. 99–111.