Parallel Algorithm for Solving Constrained Global Optimization Problems

Konstantin Barkalov and Ilya Lebedev

Lobachevsky State University of Nizhny Novgorod, Nizhny Novgorod, Russia

{[barkalov,lebedev}@vmk.unn.ru](mailto:barkalov,lebedev%7d@vmk.unn.ru)

**Abstract.** This work considers a parallel algorithm for solving multiextremal problems with non-convex constraints. The distinctive feature of this algorithm, which does not use penalty functions, is the separate consideration of each problem constraint. The search process can be conducted by reducing the original multidimensional problem to a number of related one-dimensional problems and solving this set of problems in parallel. An experimental assessment of parallel algorithm efficiency was conducted by finding the numeric solution to several hundred randomly generated multidimensional multiextremal problems with non-convex constraints.

**Keywords:** global optimization, constrained problems, non-convex constraints, dimension reduction, parallel algorithms

1. Introduction

This work considers parallel methods for solving global optimization problems with non-convex constraints. The objective function and constraints are assumed to satisfy the Lipschitz condition with a priori unknown Lipschitz constants. The analytical form of the problem’s functions may also be unknown, i.e. they can be set by an algorithm computing their values at various points within the search domain (so-called “black-box” functions). Moreover, it is supposed that even a single computation of the function value can be time-consuming, as in the applied problems it requires performing numerical simulation. These assumptions are typical for many approaches to building parallel algorithms for unconstrained global optimization [1]−[4].

At the same time, it is common for applied constrained optimization problems to be in a situation where violating one constraint results in all other functions returning indeterminate values. An example includes optimal control problems, described through systems of ordinary differential equations with a certain matrix *A* on the right side [, ]. It is only possible to calculate the optimality criteria for these problems if the matrix *A* is a Hurwitz matrix, i.e. every [eigenvalue](https://en.wikipedia.org/wiki/Eigenvalue) of *A* has [strictly negative](https://en.wikipedia.org/wiki/Negative_number) [real part](https://en.wikipedia.org/wiki/Real_part). Otherwise the value of the criteria is indeterminate.

This partial computability of the functions in constrained optimization problems substantially complicates the application of the well-known penalty function method (in some cases making it completely impossible). Thanks to its simplicity, this method is one of the most popular approaches to solving problems with constraints. However, calculating the penalty function requires first calculating the values of all of the problem’s functions at the given point, which is impossible if they are partially indeterminate.

In this work, the authors consider an approach to minimizing multiextremal functions under non-convex constraints, developed in []−[] and called the index method. The approach is based on separate considering every constraint in the problem and is not related to using penalty functions. According to the index method, each iteration (*a* *trial*) at a respective point in the search domain includes a sequential check of the problem constraints at that point. As soon as the first constraint violation is found, the trial is interrupted and the method proceeds with the next iteration; no other problem functions are calculated at that point. This allows problems to be solved in which function values may not be determined for the entire search domain. Under this approach, solving multidimensional problems is reduced (using Peano-type space-filling curves) to solving equivalent one-dimensional problems.

It should be noted that standard approaches to algorithm parallelization are not quite applicable to global optimization. For example, the rules for selecting another iteration point are quite simple and do not require parallelization (as overheads associated with organizing parallel computations will nullify any possible acceleration). Some acceleration can be achieved by parallelizing the computation of function values describing the object to be optimized; however, this approach is specific to each individual problem being solved.

The following approach looks more promising. The algorithm can be modified to run several trials in parallel. This approach provides the efficiency (as parallelization is applied to the most computation-intensive part of the problem solving process) and generality (in that it applies to a wide range of global optimization algorithms). The approach, described in [10] for unconstrained optimization, was used in this work for parallelizing constrained optimization algorithms.

The main part of the paper has the following structure. Section 2 states the constrained optimization problem, reviews the index method and an approach to reducing dimensionality by using Peano curves. Section 3 presents a parallel implementation of the index method using a set of space-filling curves. Section 4 presents the results of numerical experiments. Section 5 concludes the paper.

1. Problem Statement

Let us consider the *N*-dimensional optimization problem

(1)

(2)

The objective function (henceforth denoted by ) and the left-hand sides of the constraints satisfy Lipschitz condition

with a priory unknown constants , and may be multiextremal. It is assumed that functions are defined and computable only at the points satisfying the conditions

. (4)

Employing the continuous single-valued Peano curve mapping the unit interval [0,1] on the *x*-axis onto the *N*-dimensional domain it is possible to find the minimum in by solving the one-dimensional problem

Algorithms for numerical construction of Peano curve approximation (*evolvent*) are given in [11]. Due to the functions are defined and computable in the domains

.

These conditions allows to introduce a classification of the points according to the number of the constraints computed at this point. The index can also be defined by the conditions

where the last inequality is inessential if .

The considered dimensionality reduction scheme juxtaposes to a multidimensional problem with lipschitzian functions a one-dimensional problem, where the corresponding functions satisfy uniform Hölder condition (see [11]), i.e.,

, ,

Here is the dimensionality of the initial multidimensional problem and the coefficients are related with Lipschitz constant of the initial problem as .

Thus, a trial at a point executed at the *k*-th iteration of the algorithm will consist in the following sequence of operations.

* To determine the image in accordance with the mapping .
* To compute the values , where the index is determined by the conditions

The occurrence of the first violation of the constraint terminates the trial. In the case, when the point is a feasible one, i.e. when the trial includes the computation of the values of all functions of the problems and the index is accepted to be . The pair of values

is a *result of the trial*.

The scheme of the serial index algorithm is as follows. The first trial is executed at an arbitrary internal point . The selection of the point of any next trial is carried out by the following steps.

Step 1. Renumber the points of the preceding trials by the lower indices in increasing order of the coordinate values, i.e.

and juxtapose to them the values , , , computed at these points. The points and are introduced additionally and the values and are not defined.

Step 2. For each interval compute the *characteristics* using some formulae.

Step 3. Find the interval with the maximal characteristic

.

Step 4. Execute the next trial in the inner point of the interval , i.e. .

Step 5. Check termination condition , where is the number of interval with the maximal characteristic and is the predefined accuracy.

Detailed description of this algorithm and the corresponding theory of convergence are presented in [7]−[9].

1. Parallel Index Algorithm with the Set of Evolvents

The reduction of the multidimensional problems to the one-dimensional ones using evolvents has such important properties as the continuity and preservation of boundedness of function divided differences. However, a partial loss of information on the nearness of the points in the multidimensional space takes place since a point has only the left and the right neighbors while the corresponding point has the neighbors in directions. As a result, when using the mappings like Peano curve the images which are close to each other in the *N*-dimensional space can correspond to the preimages which can be far away from each other in the interval . This property results in the excess computations since several limit points of the trial sequence generated by the index method in the interval can correspond to a single limit point in the -dimensional space.

One of the possible ways to overcome this disadvantage consists in using the multiple mapping instead of single evolvent . To construct the set different approaches can be used. For example, in [7] a scheme was implemented, according to which each evolvent from is constructed as a result of shifting the original evolvent along the main diagonal of the hypercube *D*. The set of Peano curves thus constructed allows one to obtain from *D* for any close multidimensional images, which differ only in one coordinate, close preimages from the interval for the evolvent .

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

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

1. Determine the image for the evolvent .
2. Inform the rest of processors about the start of the trial execution at the point (*the blocking* of the point ).
3. Compute the values where the index is determined by the conditions

The occurrence of the first violation of any constraint terminates the trial at the point . In the case when is a feasible one, i.e., when , the trial includes the computation of all problem functions. In this situation, the index is set to . The triplet

is the result of the trialat the point .

1. Determine the preimages of the point and interpret the trial executed at the point as the execution of the trials in the *S* points with the same results

,

1. Inform the rest of processors about the trial results at the point .

The decision rules for the proposed parallel algorithm, in general, are the same as the rules of the sequential algorithm (except the method of the trial execution). Each processor has its own copy of the software realizing the computations of the problem functions and the decision rule of the index algorithm. For the organization of the interactions among the processors, the queues are created on each processor, where the processors store the information on the executed iterations in the form of the tuples: the processor number , the trial point , the index , and the value . Moreover, the index of the blocked point is assumed to be equal to –1; the function value at this point is undefined.

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

At every iteration of the method, the process with the index performs trials in parallel at the points . At that, each process stores all points, and an attribute indicating whether this point is blocked by another process or not is stored for each point. Let us remind that the point is blocked if the process starts the execution of a trial at this point.

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

1. Results of Numerical Experiments

A well-known approach to the investigation and comparing of the multiextremal optimization algorithms is based on testing these methods by solving a set of problems, chosen randomly from some specially designed class.

GKLS generator for the functions of arbitrary dimensionality with known properties (the number of local minima, the size of their domains of attraction, the global minimizer, etc.) has been proposed in [12]. Four GKLS classes of differentiable test functions of the dimensions *N* = 4 and 5, have been used. For each dimension, both *Hard* and *Simple* classes have been considered. The difficulty of a class was increased either by decreasing the radius of the attraction region of the global minimizer, or by decreasing the distance from the global minimizer to the domain boundaries. Application of the generator for studying some optimization algorithms has been described in [13] – [15].

In this study we will use GKLS generator to produce the constrained problems. The scheme that allows to form the constrained global optimization problems is proposed in [16]. In the previous investigations, the index method has been confirmed experimentally to be not inferior to well-known analogues. The comparing of the method to well known DIRECT one [] in solving the unconstrained optimization problems has been performed in [, ]. In the present study, an experimental investigation of the speedup, which is obtained by the use of the index method in combination with the two-level parallelization scheme from [].

The experiments have been carried out by solving a series of 100 problems with two constraints and the objective functions from the *Simple* and *Hard* GKLS classes with the dimensionalities , . The number of the used cluster nodes and, correspondingly, the number of evolvents as well as the number of cores employed at each node have been varied. The problem was considered to be solved, if the algorithm generated trial point in -vicinity of the global minimum, i.e., . The size of the vicinity was selected as , where *a* and *b* are borders of the search domain. For the purpose of simulation of the computational complexity inherent to applied problems of optimization, calculation of the problem functions in all performed experiments was made more complex by additional calculations without changing the type of function and arrangement of its minima (series summation of 80 thousand elements).

The average time and number of iterations, which were required to solve the problems of the series at various parallelization parameters are reflected in Tables 1 and 2. Here are the numbers of employed nodes and cores per a node, correspondingly.

**Table .** Average time

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Node/core* | *N*=4 | |  | *N*=5 | |
| *Simple* | *Hard* |  | *Simple* | *Hard* |
| 1/1 | 220.5 | 334.8 |  | 1223.6 | 1386.6 |
| 1/16 | 31.3 | 49.1 |  | 211.8 | 547.2 |
| 2/1 | 158.4 | 260.0 |  | 1052.9 | 1458.1 |
| 2/16 | 22.1 | 35.9 |  | 227.5 | 603.0 |
| 4/1 | 127.7 | 286.4 |  | 951.3 | 1362.2 |
| 4/16 | 20.9 | 45.0 |  | 206.0 | 925.7 |
| 8/1 | 99.3 | 141.8 |  | 700.1 | 897.3 |
| 8/16 | 31.0 | 77.7 |  | 264.6 | 374.0 |

**Table .** Average number of iterations

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Node/core* | *N*=4 | |  | *N*=5 | |
| *Simple* | *Hard* |  | *Simple* | *Hard* |
| 1/1 | 58320 | 84546 |  | 266943 | 287102 |
| 1/16 | 4297 | 6601 |  | 22655 | 56754 |
| 2/1 | 34791 | 52126 |  | 188465 | 241369 |
| 2/16 | 2029 | 3239 |  | 16689 | 40763 |
| 4/1 | 22223 | 47771 |  | 135734 | 180489 |
| 4/16 | 1281 | 2483 |  | 9241 | 35024 |
| 8/1 | 13844 | 18933 |  | 77748 | 94563 |
| 8/16 | 608 | 1473 |  | 5820 | 23033 |

The results demonstrate the presence of the speedup when using the common memory at a node (performing several trials within a problem in parallel) as well as the distributed memory (parallel solving of several subproblems at different nodes). At that, the highest time speedup was 10 (when using 64 cored on 4 cluster nodes), the highest iteration speedup was 95 (when using 128 cores on 8 cluster nodes). The difference in the speedups in time and in the number of iterations can be explained by the effect of the overheads of the data transmission between the processes. Note that when solving the applied optimization problems, the computing of the problem function values even in one point is a computation costly operation. The data transfer overheads will not affect the total computational costs predominately in this case, and the time speedup will not differ from the iteration one so strongly.

1. Conclusions

The parallel index method for solving constrained global optimization problems considered in the present work:

* allows solving the initial problem directly, without the use of the penalty functions (thus, the issues of selection the penalty coefficient and of solving a series of unconstrained problems with different penalty coefficients are eliminated);
* allows solving the problems, which the values of the problem function are not defined everywhere (for example, the objective function values are undefined out of the feasible domains of the problem constraints);
* allows using the two-level parallelization scheme with the shared and distributed memory proposed earlier for the unconstrained optimization methods.

The parallel algorithm has demonstrated speedup with respect to the number of processors/cores employed. This was confirmed by the results of the numerical solving of several hundred test problems using 128 cores of UNN computer cluster. The direction of further research is the generalization of the considered parallel algorithm for solving multicriteria problems.

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

References

1. Jones, D. R.: The direct global optimization algorithm. In: Floudas, C. A., Pardalos, P. M. (eds.) The Encyclopedia of Optimization, Second Edition. pp. 725−735. Springer, Heidelberg (2009)
2. Evtushenko, Yu.G., Malkova, V.U., Stanevichyus, A.A.: Parallel global optimization of functions of several variables. Computational Mathematics and Mathematical Physics. 49(2), 246−260 (2009)
3. Paulavicius, R., Zilinskas, J., Grothey, A.: Parallel branch and bound for global optimization with combination of Lipschitz bounds. Optimization Methods & Software 26(3), 487–498 (2011).
4. Evtushenko, Y., Posypkin, M.: A deterministic approach to global box-constrained optimization. Optimization Letters 7 (4), 819−829 (2013)
5. Balandin, D.V., Kogan, M.M.: Optimal linear-quadratic control: From matrix equations to linear matrix inequalities. Automation and Remote Control 72(11), 2276–2284 (2011)
6. Balandin, D.V., Kogan, M.M.: Pareto-optimal generalized *H*2-control and vibration isolation problems. Automation and Remote Control (2017) (article in press)
7. Strongin, R.G., Sergeyev, Ya.D.: Global optimization with non-convex constraints. Sequential and parallel algorithms. Kluwer Academic Publishers, Dordrecht (2000)
8. Sergeyev, Ya.D., Famularo, D., Pugliese, P.: Index Branch-and-Bound Algorithm for Lipschitz univariate global optimization with multiextremal constraints. J. Glob. Optim. 21(3), 317−341 (2001)
9. Barkalov, K.A., Strongin, R.G.: A global optimization technique with an adaptive order of checking for constraints. Comput. Math. Math. Phys. 42(9), 1289–1300 (2002)
10. Gergel, V., Sidorov, S.: A Two-Level Parallel Global Search Algorithm for Solution of Computationally Intensive Multiextremal Optimization Problems. In: Malyshkin, V. (Ed.) PaCT 2015, LNCS, vol. 9251, pp. 505-515. Springer, Heidelberg (2015)
11. Sergeyev, Ya.D., Strongin, R.G., Lera, D.: Introduction to global optimization exploiting space-filling curves. Springer (2013)
12. Gaviano, M., Kvasov, D.E, Lera, D., and Sergeyev, Ya.D.: Software for generation of classes of test functions with known local and global minima for global optimization. ACM Transactions on Mathematical Software 29(4), 469–480 (2003)
13. Sergeyev, Ya.D., Kvasov D.E.: Global search based on efficient diagonal partitions and a set of Lipschitz constants. SIAM J. Optim. 16(3), 910–937 (2006)
14. Paulavicius, R., Sergeyev, Y., Kvasov, D., Zilinskas, J.: Globally-biased DISIMPL algorithm for expensive global optimization. J. Glob. Optim. 59(2-3), 54–567 (2014)
15. Sergeyev, Y.D., Kvasov, D.E.: A deterministic global optimization using smooth diagonal auxiliary functions. Communications in Nonlinear Science and Numerical Simulation. 21(1-3), 99–111 (2015)
16. Gergel, V.: An Approach for Generating Test Problems of Constrained Global Optimization. In: Proceedings of Learning and Intelligent Optimization Conference (to appear)
17. Barkalov, K., Gergel, V., Lebedev, I.: Use of Xeon Phi Coprocessor for Solving Global Optimization Problems. In: Malyshkin, V. (Ed.) PaCT 2015, LNCS. 9251, 307−318 (2015)
18. Barkalov, K., Gergel, V.: Parallel global optimization on GPU. J. Glob. Optim. 66 (1), 3−20 (2016)