Comparison of several sequential and parallel derivative-free global optimization algorithms

Vladislav Sovrasov[1] (✉) and Semen Bevzuk[1]

Lobachevsky State University of Nizhni Novgorod, Russia
sovrasov.vlad@gmail.com, semen.bevzuk@gmail.com

Аннотация This work considers several stochastic and deterministic derivative-free global optimization algorithms. In the first part of the paper popular sequential open-source solvers are compared against the Globalizer solver, which is developed at the Lobachevsky State University. The Globalizer is designed to solve problems with black-box objectives satisfying the Lipschitz condition and shows competitive performance with other similar solvers. The comparison is done on several sets of challenging multi-extremal benchmark functions. The second part of this work is devoted to comparison between the Globalizer and MIDACO solvers on systems with shared and distributed memory. MIDACO is a state-of-the-art global solver included to the TOMLAB optimization environment for MATLAB. Results of the benchmark show advantages of the Globalizer on small-dimensional, but sufficiently multi-extremal benchmark functions.

Keywords: deterministic global optimization · stochastic global optimization · parallel numerical methods · derivative-free algorithms · black-box optimization

1 Introduction

Нелинейная глобальная оптимизация невыпуклых функций традиционно считается одной из самых трудных задач математического программирования. Отыскание глобального минимума функции от нескольких переменных зачастую оказывается сложнее, чем локальная оптимизация в тысячемерном пространстве. Для последней может оказаться достаточно применения простейшего метода градиентного спуска, в то время как чтобы *гаранитрованно* отыскать глобальный оптимум методам оптимизации приходится накапливать информацию о поведении целевой функции во всей области поиска [3,10,18,27]. В последнее время стали популярны различные стохастические алгоритмы глобально оптимизации, прежде всего эволюционные [12, 22, 24]. Они имеют довольно простую структуру, позволяют решать задачи большой размерности, но обеспечивают глобальную сходимость только в вероятностном смысле.

В данной работе рассмотрены open-source реализации девяти различных методов глобальной оптимизации. Все алгоритмы были протестированы на

наборе из 900 существенно многоэкстремальных функций, который был сгенерирован с помощью специализированных генераторов задач [5,7]. Помимо сравнения последовательных алгоритмов, на подмножестве из 200 тестовых задач произведено сравнение солвера MIDACO [21] и программной системы Globalizer [6] в условиях работы на суперкомпьютере Лобачевский.

2 Related Work

Ранее в литературе рассмартивалось как сравнение между собой стохастических алгоритмов глобальной оптимизации [1,17], так и детерминированных [13,14,19]. В них большинство современных методов изучены довольно детально, но упор в основном делается на последовательные алгоритмы, а основными критериями эффективности метода оптимизации является его надёжность и скорость сходимости. В большинстве работ в качестве набора тестовых функций берётся набор известных тестовых задач (например, функция Растригина, Ackley function и др.). Размер такого набора обычно не превышает 100 различных функций, некоторые из которых могут быть одноэкстремальны (как функция Розенброка).

В работе [2] софрмулированы некие общие принципы, которых, по мнению авторов, следует придерживаться при сравнении методов оптимизации. В частности, авторы говорят о преимуществах генераторов задач, позволяющих создать большие наборы задач, сводя случайные эфффекты при сравнении к минимуму. В то же время, использование одного генератора может оказаться недостаточно для исчерпывающего сравнения методов. Чтобы частично преодолеть эту проблему, авторы [2] советуют использовать несколько генераторов различной природы и создавать наборы задач различной сложности.

Учитывая опыт предыдущих работ в области сравнения методов оптимизации, в данной работе будут использованы два генератора тестовых задач разной природы, с помощью которых сгенерировано 9 наборов по 100 задач разной сложности размерности от 2 до 5. Помимо сравнения последовательных методов, также в работе приводится сравнение эффективности двух параллельных алгоритмов.

3 Statement of Multidimensional Global Optimization Problem

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

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

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

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

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

Usually, the objective function $\varphi(y)$ is defined as a computational procedure, according to which the value $\varphi(y)$ can be calculated for any vector $y \in D$ (let us further call such a calculation *a trial*). It is supposed that this procedure is time-consuming.

4 Review of Considered Optimization Methods

4.1 Parallel Algorithm of Global Search

Dimension Reduction with Evolvents 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 [23, 25, 27].

As a result of the reduction, the initial multidimensional global optimization problem (1) is reduced to the following one-dimensional problem:

$$\varphi(y(x^*)) = \min\{\varphi(y(x)) : x \in [0, 1]\}. \tag{3}$$

It is important to note that this dimensionality reduction scheme transforms the Lipschitzian function from (1) to the corresponding one-dimensional function $\varphi(y(x))$, which satisfies the uniform Hölder condition, i. e.

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

where the constant H is defined by the relation $H = 2L\sqrt{N+3}$, L is the Lipschitz constant from (2), and N is the dimensionality of the optimization problem (1).

The algorithms for the numerical construction of the Peano curve approximations are given in [27].

The computational scheme obtained as a result of the dimensionality reduction consists of the following:

- The optimization algorithm performs the minimization of the reduced one-dimensional function $\varphi(y(x))$ from (3),
- After determining the next trial point x, a multidimensional image y is calculated by using the mapping $y(x)$,
- The value of the initial multidimensional function $\varphi(y)$ is calculated at the point $y \in D$,
- The calculated value $z = \varphi(y)$ is used further as the value of the reduced one-dimensional function $\varphi(y(x))$ at the point x.

Algorithm of Global Search on Shared Memory Parallel optimization methods applied in Globalizer to solve the reduced problem (3) are based on the MAGS method, which can be presented as follows — see [25], [27].

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

Rule 1. Renumber the points of the preceding trials by the lower indices in order of increasing value of coordinates $0 = x_0 < x_1 < ... < x_{k+1} = 1$.

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

Rule 3. Determine the p intervals with the maximum characteristics $R(t_j) = \max_{1 \leq i \leq k+1} R(i)$, $j = \overline{1,p}$.

Rule 4. Execute new trials at points x^{k+j}, $j = \overline{1,p}$ located within intervals with maximum characteristics from the previous step $x^{k+j} = d(x_{t_j})$, $j = \overline{1,p}$.

The stopping condition, which terminated the trials, is defined by the inequality $\rho_{t_j} < \varepsilon$, $j = \overline{1,p}$ for the intervals with maximum characteristics from Step 3 and $\varepsilon > 0$ is the predefined accuracy of the optimization problem solution. If the stopping condition is not satisfied, the index k is incremented by p, and the new global optimization iteration is executed.

This method is employed in Globalizer to organize parallel computations on shared memory: each of p trials can be carried out on one of p local computation units.

The convergence conditions and exact formulas for descision rules $R(i)$ and $d(x)$ of the described algorithm are given, for example, in [27].

4.2 Parallel Algorithm on Distributed Memory Exploiting a Set of Evolvents

Rotated Evolvents One of the possible ways to overcome the negative effects of using a numerical approximation of evolvent (it destroys the information about the neighbor points in $\mathbb{R}^N$ space) consists in using the multiple mappings

$$Y_L(x) = \{y^0(x),\ y^1(x), ...,\ y^L(x)\} \tag{5}$$

instead of single Peano curve $y(x)$ (see [27]). The building of a set of Peano curves by rotation of the evolvents around the coordinate origin is a distinctive feature was proposed in [26]. Taking into account the initial mapping, one can conclude that current implementation of the method allows to build up to $N(N-1)+1$ evolvents for mapping the N-dimensional domain onto the corresponding one-dimensional intervals. This method for building a set of mappings can be "scaled" easily to obtain more evolvents (up to 2^N) if necessary.

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

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

on a set of intervals $[0,1]$ by the index method. Each one-dimensional problem is solved on a separate processor. The trial results at the point x^k obtained for the problem being solved by particular processor are interpreted as the results of the trials in the rest problems (in the corresponding points $x^{k_1},\ldots,x^{k_S}$). In this approach, a trial at the point $x^k \in [0,1]$ executed in the framework of the s-th problem, consists in the following sequence of operations:

1. Determine the image $y^k = y^s(x^k)$ for the evolvent $y^s(x)$.

2. Inform the rest of processors about the start of the trial execution at the point y^k (the blocking of the point y^k).

3. Determine the preimages $x^{k_s} \in [0,1], 1 \leqslant s \leqslant S$, of the point y^k and interpret the trial executed at the point $y^k \in D$ as the execution of the trials in the S points $x^{k_1},\ldots,x^{k_s}$

4. Inform the rest of processors about the trial results at the point y^k.

The decision rules for the mentioned parallel algorithm, in general, are the same as the rules of the sequential algorithm (except the method of the trial execution). Each processor has its own copy of the software realizing the computations of the problem functions and the decision rule of the index algorithm. For the organization of the interactions among the processors, the queues are created on each processor, where the processors store the information on the executed iterations in the form of the tuples: the processor number s, the trial point x^{k_s}.

The mentioned parallelization scheme was proposed in [26] and implemented in the Globalizer system with the use of MPI technology. Main features of implementation consist in the following:

- A separate MPI-process is created for each of S one-dimensional problems being solved, usually, one process per one processor employed.
- Each process can use p threads to parallel execute p trials, usually one thread per an accessible core.

4.3 MIDACO Parallel Solver

MIDACO (Mixed Integer Distributed Ant Colony Optimization) [21] is a solver which implements the evolutionary algorithm the ant colony optimization metaheuristic for continuous search domains. ACO was originally proposed to solve the TSP, but later has been successfully adopted to solve mixed integer nonlinear programming problems [22].

The MIDACO solver utilizes reverse communication architecture [21]. Reverse communication means that the call of the objective function happens outside and independently of the MIDACO source code. Relying on this feature the following distributed parallelization scheme was implemented:

1. MIDACO runs at the master node and generates $n \times p$ new trial points at each iteration.
2. Each of n distributed computational nodes gets p points from the master and performs p parallel trials on local computing devices.
3. Each of n distributed computational nodes sends p values of the objective function to the master.

In this scheme all distributed communications were implemented using the MPI library. Parallelism within a single node is powered by the OpenMP standard.

4.4 Sequential Methods

- **Algorithm of Global Search**. Sequential version of the method described in Section 4.1.
- **Locally-based Algorithm of Global Search (AGSl)** [16]. It's a modification of the original AGS which make it more locally oriented by alternately using of two types of characteristics in the Rule 2 from the Section 4.1.
- **Multi Level Single Linkage** [11]. MLSL is an improved multistart algorithm. It samples low-discrepancy starting points and does local optimizations from them. In contrast to the dummy multistart schemes MLSL uses some clustering heuristics to avoid multiple local descents to already explored local minimas.
- **DIRECT** [10]. The algorithm is deterministic and recursively divides the search space and forms a tree of hyper-rectangles (boxes). DIRECT uses the objective function values and the Lipschitz condition (2) to estimate promising boxes.
- **Locally-based DIRECT (DIRECTl)** [4]. It's a variation of DIRECT which pays less attention to non-promising boxes and therefore has less exploration power: it can converge faster on problems with few local minimas, but lost the global one in complicated cases.
- **Dual Simulated Annealing** [28]. This stochastic method is a combination of the Classical Simulated Annealing and the Fast Simulated Annealing coupled to a strategy for applying a local search on accepted locations. It converges much faster than both parent algorithms, CSA and FSA.
- **Differential Evolution** [24]. DE is an adaptation of the original genetic algorithm to the continuous search domain.
- **Controlled Random Search** [20]. The CRS starts with a set of random points and then defines the next trial point in relation to a simplex chosen randomly from a stored configuration of points. CRS in not an evolutional algorithm, although stores something like population and performs transformation resembling a mutation.
- **StoGO** [15]. StoGO is dividing the search space into smaller hyper-rectangles via a branch-and-bound approach, and searching them by a local-search algorithm, optionally including some randomness.

All the mentioned algorithms (except of AGSl are available in source codes as parts of wide-spread optimization pacages. AGS, DIRECT, DIRECTl, CRS, MLSL and StoGO are part of the NLOpt libray [8]. Differential Evolution and DSA can be found in the latest version of the SciPy [9] package for python.

5 Tools for Comparison of Global Optimization Algorithms

6 Results of Comparison

7 Conclusions

Acknowledgements

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

Список литературы

1. Ali, M.M., Khompatraporn, C., Zabinsky, Z.B.: A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. Journal of Global Optimization **31**(4), 635–672 (2005)
2. Beiranvand, V., Hare, W., Lucet, Y.: Best practices for comparing optimization algorithms. Optimization and Engineering **18**(4), 815–848 (2017)
3. Evtushenko, Y., Posypkin, M.: A deterministic approach to global box-constrained optimization. Optim. Lett. **7**, 819–829 (2013)
4. Gablonsky, J.M., Kelley, C.T.: A locally-biased form of the direct algorithm. J. Glob. Optim. **21**(1), 27–37 (2001)
5. Gaviano, M., Kvasov, D.E., Lera, D., Sergeev, 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)
6. Gergel V.P., Barkalov K.A., and Sysoyev A.V.: A novel supercomputer software system for solving time-consuming global optimization problems. Numerical Algebra, Control & Optimization **8**(1), 47–62 (2018)
7. Grishagin, V.: Operating characteristics of some global search algorithms. Problems of Statistical Optimization **7**, 198–206 (1978 (In Russian))
8. Johnson, S.G.: The nlopt nonlinear-optimization package. URL http://ab-initio.mit.edu/nlopt. [Online; accessed <24.12.2018>]
9. Jones, E., Oliphant, T., Peterson, P., et al.: SciPy: Open source scientific tools for Python (2001–). URL http://www.scipy.org/. [Online; accessed <24.12.2018>]
10. Jones, D.R.: The direct global optimization algorithm. In: The Encyclopedia of Optimization, pp. 725–735. Springer, Heidelberg (2009)
11. Kan, A.H.G.R., Timmer, G.T.: Stochastic global optimization methods part ii: Multi level methods. Math. Program. **39**, 57–78 (1987)
12. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, pp. 1942–1948 vol.4 (1995)
13. Kvasov, D.E., Mukhametzhanov, M.S.: Metaheuristic vs. deterministic global optimization algorithms: The univariate case. Applied Mathematics and Computation **318**, 245 – 259 (2018)
14. Liberti, L., Kucherenko, S.: Comparison of deterministic and stochastic approaches to global optimization. International Transactions in Operational Research **12**, 263 – 285 (2005)

15. Madsen, K., Zertchaninov, S.: A new branch-and-bound method for global optimization (1998)
16. Markin, D.L., Strongin, R.G.: A method for solving multi-extremal problems with non-convex constraints, that uses a priori information about estimates of the optimum. Computational Mathematics and Mathematical Physics **27(1)**, 33–39 (1987)
17. Mullen, K.: Continuous global optimization in r. Journal of Statistical Software, Articles **60**(6), 1–45 (2014)
18. Paulavivcius, R., Zilinskas, J., Grothey, A.: Parallel branch and bound for global optimization with combination of lipschitz bounds. Optim. Methods Softw. **26**(3), 487–498 (1997)
19. Pošík, P., Huyer, W., Pál, L.: A comparison of global search algorithms for continuous black box optimization. Evolutionary Computation **20**(4), 509–541 (2012)
20. Price, W.L.: Global optimization by controlled random search. Journal of Optimization Theory and Applications **40**(3), 333–348 (1983)
21. Schlueter, Martin: Nonlinear mixed integer based optimization technique for space applications. Ph.D. thesis, The University of Birmingham (2012)
22. Schluter, M., Egea, J.A., Banga, J.R.: Extended ant colony optimization for non-convex mixed integer nonlinear programming. Computers & Operations Research **36**(7), 2217 – 2229 (2009)
23. Sergeyev, Y.D., Strongin, R.G., Lera, D.: Introduction to Global Optimization Exploiting Space-Filling Curves. Springer Briefs in Optimization, Springer, New York (2013)
24. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization **11**(4), 341–359 (1997)
25. Strongin, R.: Numerical Methods in Multiextremal Problems (Information-StatisticalAlgorithms). Moscow: Nauka (In Russian) (1978)
26. Strongin, R.G., Gergel, V.P., Barkalov, K.A.: Parallel methods for global optimization problem solving. Journal of instrument engineering **52**, 25–33 (2009 (In Russian))
27. Strongin R.G., Sergeyev Ya.D.: Global optimization with non-convex constraints. Sequential and parallel algorithms. Kluwer Academic Publishers, Dordrecht (2000)
28. Xiang, Y., Sun, D., Fan, W., Gong, X.: Generalized simulated annealing algorithm and its application to the thomson model. Physics Letters A **233**(3), 216 – 220 (1997)