Comparison of Several Stochastic and Deterministic Derivative-free Global Optimization Algorithms*

Vladislav Sovrasov

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

Abstract. In this paper popular open-source solvers are compared against Globalizer solver, which is developed at the Lobachevsky State University. The Globalizer is designed to solve problems with black-box objective function 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. Also this work considers a method of heuristic hyperparameters control for the Globalizer allowing to reduce amount of initial tuning before optimization. The proposed scheme allows substantially increase convergence speed of the Globalizer by switching between "local" and "global" search phases in runtime.

Keywords: deterministic global optimization · stochastic global optimization · algorithms comparison · derivative-free algorithms · black-box optimization · multi-extremal problems

1 Introduction

The problem of finding the global minima of the nonlinear nonconvex functions is considered to be one of the most difficult mathematical programming problems traditionally. Often, it appears to be more complex than the local optimization in an essentially multidimensional space. For the latter, the application of the simplest gradient descent method or of the pattern search algorithms may appear to be sufficient [26] whereas in order to *guarantee* the finding of the global optimum, the optimization methods have to accumulate the information on the behavior of the objective function in the whole search domain [3, 10, 17, 25]. Recently, various stochastic global optimization algorithms, first of all, the evolution ones [12, 20, 23] became popular. These ones have rather simple structure and allow solving the problems of large dimensionality. However, these methods provide the global convergence in the probabilistic meaning only.

In the present work, the open-source implementations of the eight different global optimization methods included into the NLOpt library [8] and the SciPY

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

package [9] are considered. All algorithms were tested on a set of 900 essentially multiextremal functions, which has been generated with the use of special problem generators [5, 7].

2 Related Work

Earlier, the comparison of the stochastic global optimization algorithms [1, 16] as well as of the deterministic ones [13, 14, 18] between each other has been considered in the literature. In these works, most of modern methods have been studied in details. In the majority of works, the sets of well-known test problems (for example, the Rastrigin function, Ackley function, etc.) were taken as the sets of test functions. The sizes of such sets don't exceed 100 different functions usually, some of which can be the single-extremal ones (such as the Rosenbrock function).

In [2], some general principles were formulated, which, in the author's opinion, should be obeyed when comparing the optimization methods. In particular, the authors say about the advantages of the problem generators allowing generating the large sets of problems thus minimizing the random effects when comparing the methods. At the same time, the use of a single generator can appear to be not enough for a comprehensive comparison of the methods. In order to overcome this problem in part, the authors of the paper [2] advise to use several generators of various nature and to create the sets of problems of various complexity.

Taking into account the experience of the preceding works in the field of comparison of the optimization methods, two generators of the test problems of different nature will be used in the present work. Using these ones, 9 sets of 100 problems of various complexity with the dimensionality varying from 2 to 5 were generated.

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:

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

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

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

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 Algorithm of Global Search

Dimension Reduction with Space-Filling Curves 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 [22,24,25].

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 [25].

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.

Optimization method applied in Globalizer [6] to solve the reduced problem (3) is based on the AGS method, which can be presented as follows — see [24], [25].

The algorithm considered for solving the stated problem implies generating a sequence of points x_k, in which the values of the minimized function $z_k =$

$f(x_k)$ are computed. Let us call the process of computating the function value (including calculating an image $y^k = y(x^k)$) a trial, and the pair (x^k, z^k) — the result of the trial. A set of the pairs $\{(x^k, z^k)\}, 1 \leqslant k \leqslant n$ makes up the search information accumulated by the method after executing n steps.

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.

Step 1. Renumber the points in the set $X_k = \{x^1, \ldots, x^k\} \cup \{0\} \cup \{1\}$, which includes the boundary points of the interval $[0, 1]$ as well as the points of preceding trials, by the lower indices in order of increasing coordinate values i.e.

$$0 = x_0 < x_1 < \ldots < x_{k+1} = 1$$

Step 2. Assuming $z_i = f(x_i), 1 \leqslant i \leqslant k$, compute the values

$$\mu = \max_{1 \leqslant i \leqslant k} \frac{|z_i - z_{i-1}|}{\Delta_i}, M = \begin{cases} r\mu, \mu > 0 \\ 1, \mu = 0 \end{cases} \tag{5}$$

where $r > 1$ is a predefined parameter for the method, and $\Delta_i = (x_i - x_{i-1})^{\frac{1}{N}}$.

Step 3. For each interval $(x_{i-1}, x_i), 1 \leqslant i \leqslant k+1$, compute the characteristics according to the formulae

$$R(1) = 2\Delta_1 - 4\frac{z_1}{M}, R(k+1) = 2\Delta_{k+1} - 4\frac{z_k}{M}, \tag{6}$$

$$R(i) = \Delta_i + \frac{(z_i - z_{i-1})^2}{M^2 \Delta_i} - 2\frac{z_i + z_{i-1}}{M}, 1 < i < k+1. \tag{7}$$

Step 4. Determine the interval with the maximum characteristic $(x_{t-1}, x_t), t = \arg\max_{1 \leqslant i \leqslant k+1} R(i)$

Step 5. Execute a new trial at point x_{k+1} computed according to the formula

$$x_{k+1} = \frac{x_t + x_{t-1}}{2}, t = 1, t = k+1,$$

$$x_{k+1} = \frac{x_t + x_{t-1}}{2} - \operatorname{sign}(z_t - z_{t-1})\frac{1}{2r}\left[\frac{|z_t - z_{t-1}|}{\mu}\right]^N, 1 < t < k+1. \tag{8}$$

The stopping condition, which terminated the trials, is defined by the inequality $\Delta_t \leqslant \varepsilon$ for the interval with the maximum characteristics from Step 4 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 1, and the new global optimization iteration is executed.

The convergence conditions of the described algorithm are given, for example, in [25].

Hyperparameters control in AGS The parameter r from (5) affects the global convergence of AGS directly (see [25], Chapter 8): at high enough value of r, the method converges to all global minima of the objective function with guarantee. At the same time, according to (7) and (8), at the infinitely high value of r, AGS turns into the brute force search method on a uniform grid.

In the ideal case, in order to provide the highest convergence speed, the estimate of the Lipschitz constant from (5) should not be too overestimated, but in practice the actual value of L from (2) in unknown, and one has either to take an obviously overestimated value of r or to execute several runs of AGS with different parameters. In order to resolve the problem of choosing r to some extent, let us use the following scheme:

- execute q iterations of AGS with $r = r_{max}$;
- execute q iterations of AGS with $r = r_{min}$;
- repeat the above steps either until convergence or until the allowed number of iterations are exhausted.

In the above algorithm, $r_{min} < r_{max}$, $q > 1$. Instead one parameter r, now 3 ones should be selected. However, according to the results of the numerical experiments, it is easier than to find the optimal value of r. Intuitively, the practical efficiency of the proposed scheme can be explained by the fact that now the operation of the method takes place in two modes: the global search with $r = r_{max}$ and the local one with $r = r_{min}$. If during the global search phase, the method approached the global minimum whereas during the next phase, the estimate of the global minimum would be refined rapidly. If two phases are not enough, the process is continued. This way, a better trade-off between the exploration and the exploitation is achieved. Further, we will denote the method utilizing the scheme described above as AGS-AR.

4.2 Other Optimization Methods

- **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 minima.
- **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-biased 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 minima, but lost the global one in complicated cases.
- **Dual Simulated Annealing** [27]. 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** [23]. DE is an adaptation of the original genetic algorithm to the continuous search domain.
- **Controlled Random Search** [19]. 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 are available in source codes as parts of widespread optimization packages. DIRECT, DIRECTl, CRS, MLSL and StoGO are part of the NLOpt library [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

The use of the sets of test problems with known solutions generated by some random mechanisms is one of commonly accepted approaches to comparing the optimization algorithms [2]. In the present work, we will use two generators of test problems generating the problems of different nature [5, 7] [1].

Let us denote the problem set obtained with the use of the first generator from [7] as F_{GR}. The mechanism of generation of the problems F_{GR} doesn't provide the control of the problem complexity and of the number of local optima. However, the generated functions are known to be the multiextremal ones essentially. Besides, the problems generated by F_{GR} are the two-dimensional ones. In the present work, we will use 100 functions from the class F_{GR} generated randomly.

The GKLS generator [5] allows obtaining the problems of given dimensionality with given number of extrema. Moreover, GKLS allows adjusting the complexity of the problems by decreasing or increasing the size of the global minimum attractor. In [21] the parameters of the generator allowing generating the sets of 100 problems each of two levels of complexity (Simple and Hard) of the dimensionality equal to 2, 3, 4, and 5 are given. Following the authors of the GKLS generator, we will use the parameters proposed by them and, this way, add 800 more problems of various dimensionalities and complexity into the test problem set.

Let us suppose a test problem to be solved if the optimization method executes the scheduled trial y^k in a δ-vicinity of the global minimum y^*, i.e.

[1] Software implementations of these generators are available in source codes at the page https://github.com/sovrasov/global-optimization-test-problems

$\left\|y^k - y^*\right\| \leq \delta = \alpha \left\|b - a\right\|$, where a and b are the left and the right boundaries of the hypercube from (1), α is relative precision. If this relation is not fulfilled before the expiration of the limit of the number of trials, the problem was considered to be unsolved. The limit of the number of trials and α were set for each problem class according to the dimensionality and complexity (see Table 1).

Table 1: Trials limits and relative precision for the test problem classes

Problems class	Trials limit	α
F_{GR}	5000	0.01
GKLS 2d Simple	8000	0.01
GKLS 2d Hard	9000	0.01
GKLS 3d Simple	15000	0.01
GKLS 3d Hard	25000	0.01
GKLS 4d Simple	150000	$\sqrt[4]{10^{-6}}$
GKLS 4d Hard	250000	$\sqrt[4]{10^{-6}}$
GKLS 5d Simple	350000	$\sqrt[5]{10^{-7}}$
GKLS 5d Hard	600000	$\sqrt[5]{10^{-7}}$

Let us consider the averaged number of trials executed to solve a single problem and the number of solved problems as the characteristics of the optimization method on each class. The less the number of trials, the faster the method converges to a solution, hence the less times it turns to a potentially computation-costly procedure of computing the objective function. The number of solved problems evidences the reliability of the method at given parameters on the class of test problems being solved. In order to make independent the quantities featuring the reliability and the speed of convergence, averaged number of trials always was calculated taking into account solved problems only.

The average number of trials doesn't represent the real behavior of an optimization method on a problems set in some cases. For an instance, if a method performs well on the most problems and spends too much trials to solve the least several problems, we wouldn't catch such case looking at the average number of trials only. As an advanced measure of performance we will use the operating characteristic [7]. It's defined by a set of points on the (K, P) plane where K is the average number of search trials conducted before satisfying the termination condition when minimizing a function from a given class, and P is the proportion of problems solved successfully. If at a given K, the operating characteristic of a method goes higher than one from another method, it means that at fixed search costs, the former method has a greater probability of finding the solution. If some value of P is fixed, and the characteristic of a method goes to the left from that of another method, the former method requires fewer resources to achieve the same reliability.

6 Results of Numerical Experiments

The results of various algorithms on different problem classes depend on the adjustments of algorithms directly. In most cases, the authors of software implementations are oriented onto the problems of medium difficulty. In order to obtain a satisfactory result when solving the essentially multiextremal problems, a correction of some parameters is required. When conducting the comparison, the following parameters for the methods were employed:

- in the AGS-AR method, the parameter of alternation the global and local stages q was set to be equal to $50 \cdot \log_2(N-1) \cdot N^2$, also $r_{min} = 3$, $r_{max} = 2 \cdot r_{min}$;
- in the DIRECT and DIRECTl methods, the parameter $\epsilon = 10^{-4}$;
- in the SDA method, the parameter $visit = 2.72$.

The rest parameters were varied subject to the problem class (see Table 2). For the AGS the value ot the r parameter, such that the method solves all problems and performs the minimum amount of trials, was estimated by brute force on the uniform grid with step 0.1.

Table 2: Class-specific parameters of the optimization algorithms

	AGS	CRS	DE
F_{GR}	$r = 3$	popsize=150	mutation=(1.1,1.9), popsize=60
GKLS 2d Simple	$r = 4.6$	popsize=200	mutation=(1.1,1.9), popsize=60
GKLS 2d Hard	$r = 6.5$	popsize=400	mutation=(1.1,1.9), popsize=60
GKLS 3d Simple	$r = 3.7$	popsize=1000	mutation=(1.1,1.9), popsize=70
GKLS 3d Hard	$r = 4.4$	popsize=2000	mutation=(1.1,1.9), popsize=80
GKLS 4d Simple	$r = 4.7$	popsize=8000	mutation=(1.1,1.9), popsize=90
GKLS 4d Hard	$r = 4.9$	popsize=16000	mutation=(1.1,1.9), popsize=100
GKLS 5d Simple	$r = 4$	popsize=25000	mutation=(1.1,1.9), popsize=120
GKLS 5d Hard	$r = 4$	popsize=30000	mutation=(1.1,1.9), popsize=140

The results of running the optimization methods on the considered problem classes are presented in Tables 3, 4. The DIRECT, AGS and AGS-AR methods have demonstrated the best convergence speed on all classes, at that AGS-AR inferior to DIRECT on the 2d problems from the Simple classes and has an advantage on the problems of the Hard classes. As one can see from Table 4, the deterministic methods (AGS, AGS-AR, DIRECT, and DIRECTl) were the most reliable. Among the stochastic methods, MLSL and SDA have demonstrated the highest reliability.

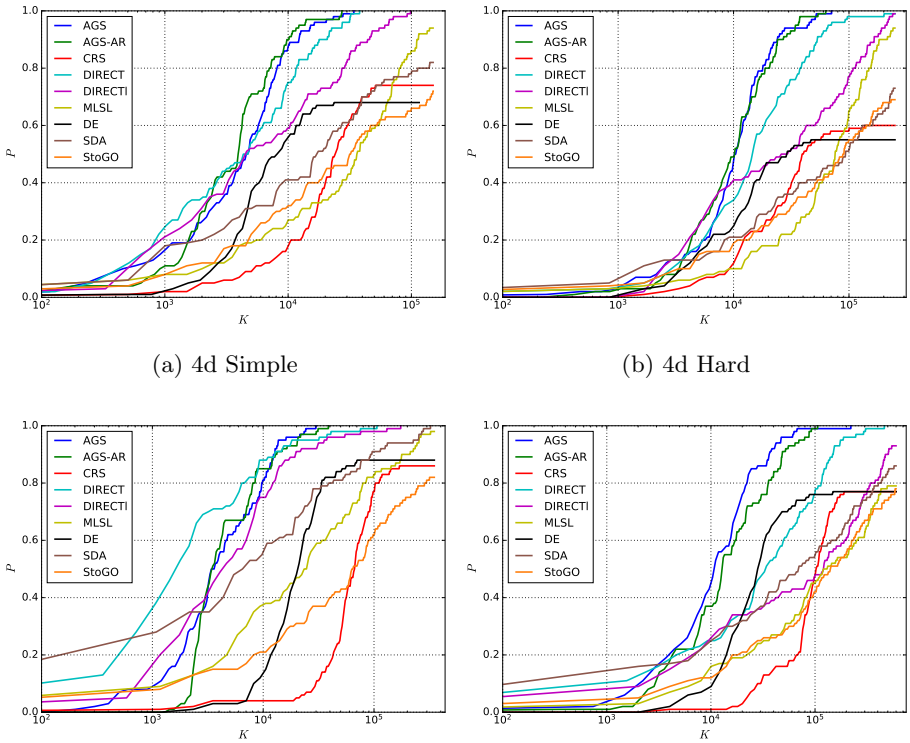
Operating characteristic of the methods (Figures 1a, 1b, 1c, 1d) demonstrates that AGS and AGS-AR faster than the other methods achieve 100% success rate. Also on GKLS 5d Simple the DIRECT generally has the best performance, but there are several hard problems that affect it's average number of trials metric.

Table 3: Averaged number of trials executed by optimization methods for solving the test optimization problems

	AGS	AGS-AR	CRS	DIRECT	DIRECTl	MLSL	SDA	DE	StoGO
F_{GR}	193.1	248.3	400.3	**182.2**	214.9	947.2	691.2	1257.3	1336.8
GKLS 2d Simple	254.9	221.6	510.6	**189.0**	255.2	556.8	356.3	952.2	1251.5
GKLS 2d Hard	**728.7**	785.0	844.7	985.4	1126.7	1042.5	1637.9	1041.1	2532.2
GKLS 3d Simple	1372.1	1169.5	4145.8	**973.6**	1477.8	4609.2	2706.5	5956.9	3856.1
GKLS 3d Hard	3636.1	**1952.1**	6787.0	2298.7	3553.3	5640.1	4708.4	6914.3	7843.2
GKLS 4d Simple	5729.8	**4919.1**	19883.6	7328.8	15010.0	41484.8	22066.0	6271.2	29359.2
GKLS 4d Hard	13113.4	**12860.1**	27137.4	22884.4	55596.1	80220.1	68048.0	12487.6	58925.5
GKLS 5d Simple	**5821.5**	6241.3	62921.7	5966.1	10795.5	52609.2	34208.8	20859.4	69206.8
GKLS 5d Hard	**17008.6**	21555.1	87563.9	61657.3	148637.8	138011.8	115634.6	26850.0	141886.5

Table 4: Number of test optimization problems solved by the methods

	AGS	AGS-AR	CRS	DIRECT	DIRECTl	MLSL	SDA	DE	StoGO
F_{GR}	100	100	76	100	100	97	96	96	67
GKLS 2d Simple	100	100	85	100	100	100	100	98	90
GKLS 2d Hard	100	97	74	100	100	100	93	85	77
GKLS 3d Simple	100	100	75	100	100	100	89	86	44
GKLS 3d Hard	100	100	72	100	99	100	88	77	43
GKLS 4d Simple	100	100	74	100	100	94	82	68	72
GKLS 4d Hard	100	100	60	99	99	94	73	55	69
GKLS 5d Simple	100	100	86	100	100	98	100	88	82
GKLS 5d Hard	100	100	77	100	93	79	86	77	78



(a) 4d Simple (b) 4d Hard

(c) 5d Simple (d) 5d Hard

Fig. 1: Operating characteristics of the algorithms when solving problems from the GKLS 4d and 5d classes. Best viewed in color.

Robustness of AGS and AGS-AR to the Hyperparameters Choice. In order to investigate the influence of hyperparameters to the convergence speed of the AGS and AGS-AR, experiments with the following settings were conducted on the problems from GKLS 5d Simple class:

- AGS with $r = 4$ (like in the Table 2);
- AGS with $r = 6$;
- AGS-AR with parameters from the beginning of the Section 6 ($q = 50 \cdot \log_2(4) \cdot 25 = 2500$, $r_{min} = 3$, $r_{max} = 2 \cdot r_{min}$);
- AGS-AR with $r_{max} = 8$ and other parameters from the previous experiment;
- AGS-AR with $q = 1000$ and other parameters from the beginning of the Section 6;

The operating characteristics collected in the experiments described above are shown in the Figure 2. AGS with $r = 6$ (the cyan-colored curve) shows the worst convergence speed, which indicates that AGS is very sensitive to choice

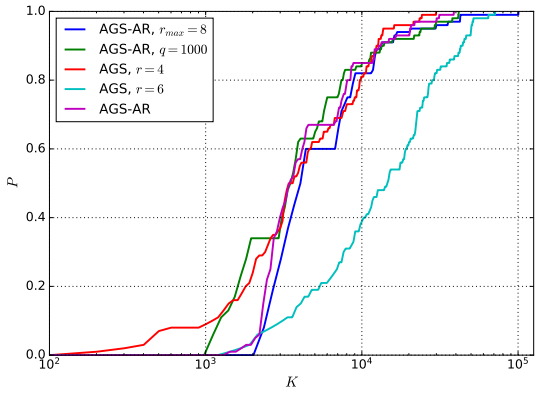


Fig. 2: Operating characteristics of AGS and AGS-AR with different hyperparameters when solving problems from the GKLS 5d Simple classe. Best viewed in color.

of r. Since on the start AGS-AR has the same value of r as AGS with $r = 6$, operating characteristics of these methods are identical up to $K = 2500$. After that point AGS-AR switches to $r = 3$ and rapidly begins to increase the amount of solved problems until the next exploration phase on $K = 5000$. Intervals where AGS-AR works with $r = r_{max}$ are visible on the operating characteristics as plateaus. Variations of r and q didn't drastically change the operating characteristic of AGS-AR. The latter observation shows robustness of the proposed AGS modification with the alternating parameter r.

7 Conclusions

In the present paper, several global optimization algorithms were considered. A comparison of efficiencies of these ones has been done on a set of test problems. Also a scheme of hyperparameters control for the AGS algorithms was proposed and evaluated. The results presented in this work allow making the following conclusions:

- the proposed modification of the stock AGS, AGS-AR allows to pay less attention to initial hyperparameter tuning and performs on-par with properly tuned AGS;
- AGS-AR method has demonstrated the convergence speed and reliability at the level of DIRECT and exceeds many other algorithms, the open-source implementations of which are available;
- the stochastic optimization methods inferior to the deterministic ones in the convergence speed and in reliability. It is manifested especially strongly on more complex multiextremal problems.

References

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 Stochastic Search **7**, 198–206 (In Russian) (1978)
8. Johnson, S.G.: The nlopt nonlinear-optimization package, 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–), 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. Mullen, K.: Continuous global optimization in r. Journal of Statistical Software, Articles **60**(6), 1–45 (2014)
17. 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)
18. Pok, P., Huyer, W., Pl, L.: A comparison of global search algorithms for continuous black box optimization. Evolutionary Computation **20**(4), 509–541 (2012)
19. Price, W.L.: Global optimization by controlled random search. Journal of Optimization Theory and Applications **40**(3), 333–348 (1983)
20. 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)

21. Sergeyev, Y., Kvasov, D.: Global search based on efficient diagonal partitions and a set of Lipschitz constants. SIAM Journal on Optimization **16**(3), 910–937 (2006)
22. Sergeyev, Y.D., Strongin, R.G., Lera, D.: Introduction to Global Optimization Exploiting Space-Filling Curves. Springer Briefs in Optimization, Springer, New York (2013)
23. 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)
24. Strongin, R.: Numerical Methods in Multiextremal Problems (Information-StatisticalAlgorithms). Moscow: Nauka (In Russian) (1978)
25. Strongin R.G., Sergeyev Ya.D.: Global optimization with non-convex constraints. Sequential and parallel algorithms. Kluwer Academic Publishers, Dordrecht (2000)
26. Torczon, V.: On the convergence of pattern search algorithms. SIAM Journal on Optimization **9**(1), 1–25 (1997)
27. 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)