Lipschitz global optimization and machine learning: helping each other to solve complex problems

Marina Usova*, and *Konstantin* Barkalov

Lobachevsky State University of Nizhny Novgorod, 603022, Nizhny Novgorod, Russia

Abstract. In this paper we consider global optimization problems and methods for solving them. The numerical solution of this class of problems is computationally challenging. The most complex problems are multicriteria problems in which the objective functions are multiextremal and non-differentiable, and, moreover, given in the form of a "black box", i.e. calculating the objective function at a point is a time-consuming operation. Particularly, we consider an approach to acceleration of the global search using machine learning methods. At the same time, the problem of tuning the hyperparameters of the machine learning methods themselves is very important. The quality of machine learning methods is substantially affected by their hyperparameters, while the evaluation of the quality metrics is a time-consuming operation. We also consider an approach to hyperparameter tuning based on the Lipschitz global optimization. These approaches are implemented in the iOpt open-source framework of intelligent optimization methods.

1 Introduction

Global optimization problems often arise in the problem of parameters identification in mathematical modeling of processes. Among the candidate models, the model that best fits the experimental data is searched. In this case, the objective function is a criterion for assessing the degree of correspondence of the tested model to experimental data. The numerical solution of such problems is associated with issues of dimension and type of objective function. Multiextremal optimization problems have a significantly higher complexity compared to other types of optimization problems, because the search for a global optimizer requires exploration of the entire search domain. When using uniform grid search, the computational costs for solving such problems grow exponentially with increasing dimension.

In such problems the objective function is usually given in the form of a "black box" [1]. This term reflects the complete lack of information about the internal structure of the function. It is specified in the form of a certain computational procedure that has an "input" for a function argument and an "output" for the value of the function. In applied global optimization problems, each calculation of the objective function value is a very labor-

* Corresponding author: usova@itmm.unn.ru

intensive operation, therefore the construction of efficient algorithms for finding the optimal solution involves reducing the number of such calculations.

The possibility of finding a global minimum in such problems is fundamentally based on the availability of a priori information that allows one to relate the position of global optimizer of the objective function with known values at the points of the search trials performed. The search trial is the calculation of the objective function values at a given point. In this sense, one of the natural and important (both in theory and practice) assumptions is the supposition that the objective function satisfies the Lipschitz condition with a priori unknown constant L [2, 3]. These complex problems and methods for solving them is considered in the paper.

To solve global optimization problems, a number of efficient deterministic algorithms that guarantee convergence to a global minimum are known [3-8]. A number of metaheuristic algorithms based on the random search concept are also widely known [9-10].

An efficient method for solving this class of problems is the global search algorithm (GSA) [11, 12], based on the ideas of purposeful selection of the next trial point by cutting off non-prospective search subregions and exploring only those that may contain a solution to the problem.

With the rapid increase in the complexity of the models being studied, classical optimization algorithms are becoming less and less applicable. Machine learning has become the most commonly used method for solving larger and more complex problems. This leads researchers to combine known optimization approaches with machine learning methods, taking into account the specifics of the problems being solved [13-16].

At the same time, machine learning methods themselves often need help. In recent years, hyperparameter tuning has become an area of intense research, as proper hyperparameter selection determines the practical applicability of machine learning methods [17-19]. This paper discusses an approach to accelerating the global search algorithm in multicriteria problems using machine learning methods, as well as the problem of effectively tuning the hyperparameters of machine learning methods.

In the first case, the use of *decision trees* [20] makes it possible to combine GSA with the local optimization method. The use of decision trees built on the basis of accumulated search information allows one to make a decision about launching a local method in the detected region of attraction of a local minimum. Such a combination can reduce the number of trials performed, and therefore speed up the convergence of the method.

In the second case, GSA is used as a mechanism to help the machine learning methods themselves. Since quality of ML methods is significantly influenced by the hyperparameters, and the evaluation of quality metrics is a labor-intensive operation, hyperparameters tuning is a classic area of application for global optimization methods. To select hyperparameters, along with the well-known Bayesian optimization [21-22], Lipschitz global optimization methods, including GSA, can be used.

The mentioned approaches are implemented in the open source framework of intelligent optimization methods iOpt. The paper provides brief description of the approaches, as well as experimental results demonstrating their efficiency.

2 Problem statement

Let's consider the problem of finding the global minimum of a function $\varphi(y)$ in the hyperinterval D

$$\varphi^* = \varphi(y^*) = \min_{y \in D}\{\varphi(y)\}, \tag{1}$$

where $D = \{ y \in R^N : a_i \le y_i \le b_i, a, b \in R^N, 1 \le i \le N\}$, and a, b are given vectors.

We will assume that the function $\varphi(y)$ satisfies the Lipschitz condition

$$|\varphi(y') - \varphi(y'')| \leq L\|y' - y''\|, \; y', y'' \in D, \tag{2}$$

where L is a priory unknown Lipschitz constant. Thus, a limited change in the argument Δy generates a limited change in the function values $\Delta\varphi$. This assumption can be interpreted (in relation to applied problems) as a reflection of the limited power that generates changes in the simulated system.

The Lipschitz condition allows a clear geometric interpretation: the graph of the function on the interval $[y', \; y'']$ should be located inside the area limited by the lines passing through the points $(y', \varphi(y'))$ and $(y'', \varphi(y''))$ with the angular coefficients L and $-L$ (see Fig. **1**).

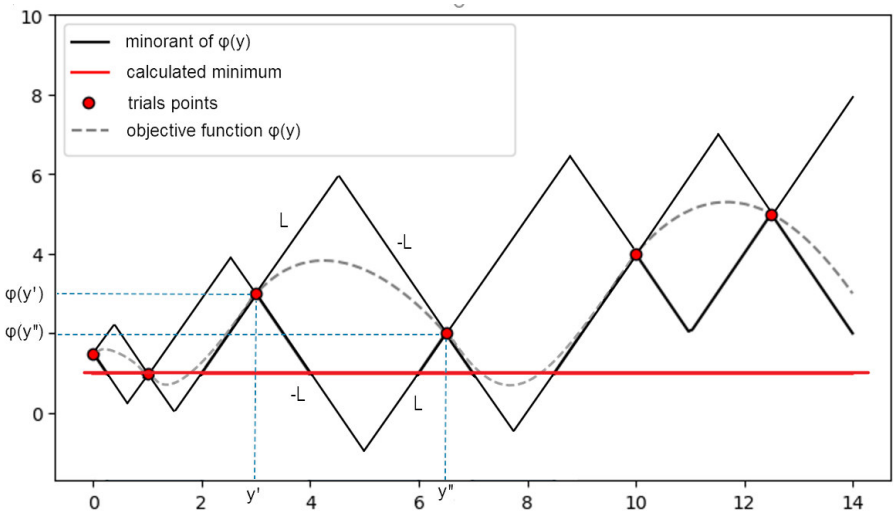

Fig. 1. Geometric meaning of the Lipschitz condition.

The objective function $\varphi(y)$ can be given in the form of a "black box", as a result of the work of some subroutine or library.

The numerical solution of problem (1) assumes constructing an estimate $y_k^* \in D$ based on a finite number k of calculations of the objective function values. The constructed estimate is considered a solution to the problem if $||y^* - y_k^*|| \leq \varepsilon$, where $\varepsilon \geq 0$ is the given accuracy.

On the other hand, the problem of hyperparameters tuning can be considered as a global optimization problem of the form (1), where $y \in R^N$ is the parameters vector, $\varphi(y)$ is the objective function, D is the feasible domain. It is assumed that in domain D the objective function $\varphi(y)$ is multiextremal.

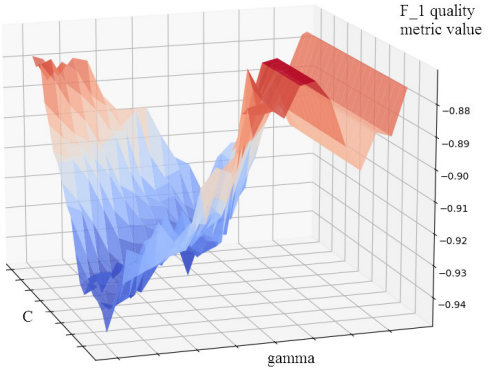

Fig. 2. The F_1 score for the support vector classification (SVC) depending on the hyperparameters.

An example of such problem is the task of tuning the hyperparameters of a classification algorithm to build a breast cancer prediction model. Figure 2 shows the graph of the quality metrics of the model depending on the tuned parameters. The graph shows that the objective function is non-convex, and the choice of parameters can significantly affect the prediction quality.

3 Algorithms

3.1 General scheme of divide-the-best algorithms

A number of general schemes for global optimization algorithms with a similar structure have been proposed in the literature. For example, branch and bound methods [23-24], characteristic methods [25], adaptive partition algorithms [26], described and studied within the framework of unified theory. Most of these global optimization algorithms can be classified into a more general class of methods – *divide-the-best* algorithms [27]. The convergence conditions for these algorithms have been studied in detail, see [8].

The idea of these methods is to construct a so-called *characteristic* for each subregion of the search domain. The characteristic quantitatively assesses the possibility of finding the global minimum point within the considered subregion and depends on the results of all trials performed. The general computational scheme of this type of algorithm is shown in Figure 3.

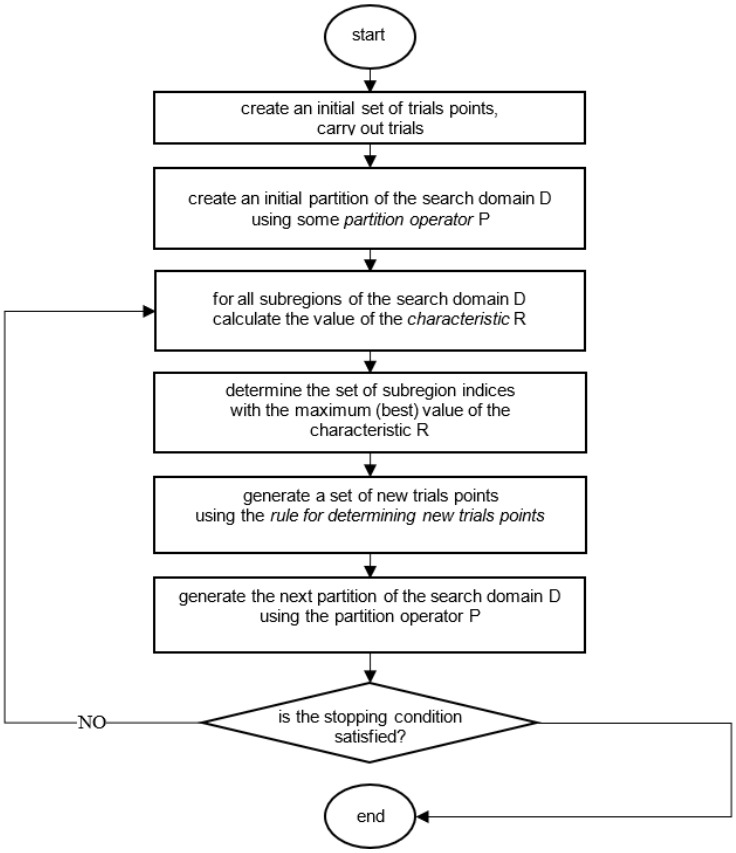

Fig. 3. Computational scheme of divide-the-best algorithms.

The *global search algorithm* (GSA) used in this study belongs to the class of divide-the-best algorithms. The rules of the algorithm are constructed in such a way that the next trial carried out in the interval with the maximum characteristic, namely at the global minimizer of the current constructed minorant of the objective function (see Fig. 1**).**

When solving multidimensional problems, dimensionality is reduced using Peano curves. They allow one to reduce the problem of searching the minimum of the Lipschitz function $\varphi(y)$ to a one-dimensional problem with the objective function $\psi(x)$, that satisfies the Hölder condition

$$|\psi(x') - \psi(x'')| \leq K|x' - x''|^{1/N},\ x', x'' \in [0,1], \quad (3)$$

where N is the dimension of the original multidimensional problem, and the coefficient K is related to the Lipschitz constant L by the formula $K \leq 2L\sqrt{N+3}$ [12].

This approach can also be generalized to the case of parallel computing. Thus, at each iteration of the method, $p > 1$ trials are carried out in parallel. Their results are entered into the information base, and the algorithm proceeds to calculating new p trial points.

The complete description of the algorithm is available, for example, in [12]. For a rationale for organizing parallel computing, see [28]. Modifications that take into account the inequality constraints are presented in [29-30].

3.2 Running a local method as a search acceleration tool

Speeding up global search algorithms can be achieved by combining fast local algorithms with slower global search. The idea is to start a fast local search from some trial point belonging to the attraction region of the local extremum.

At the first stage of the search, the information is accumulated in the form of trial results. At the second stage, the accumulated information is used to determine whether a point belongs to the attraction region. Upon completion of the local search, the global algorithm continues its work, and the end point of the local descent is added to the search information database for the global search, while intermediate points of the descent trajectory are used only to refine the current estimate of the Lipschitz constant.

We use the Hooke-Jeeves local search method for the class of problems under consideration. This algorithm belongs to the class of zero-order methods. Note that for the problems considered with a non-differentiable objective function specified in the form of a black box, gradients are too expensive to calculate [31]; moreover, the gradient value can be calculated with an error. Detailed description of Hooke-Jeeves method is available in [32].

Launching such a local method from attraction regions of local minima during the global search will reduce the number of trials of the algorithm, and therefore speed up its operation. In this case, the decision to launch a local method can be made using decision trees that are classic for machine learning.

3.3 Function approximation using decision trees

The decision to run the local method is based on a comparison of neighboring trial points. This comparison can be made only in a multidimensional space, and not on a one-dimensional interval. This is due to changes in the properties of the problem with reduced dimension. In particular, after dimension reduction, one local minimum in a multidimensional space will correspond to many minima on one-dimensional interval.

Determining neighboring points in a multidimensional space is a rather labor-intensive task, therefore we proposed to use the function approximation that can be obtained using decision trees.

In general, a decision tree is a hierarchical decision-making scheme in the form of a graph. A decision tree is a machine learning tool used to automatically analyze large amounts of

data. The nodes of such a tree contain simple decision-making rules; in most cases, they check the value of one parameter. Depending on the decisions made, a path is selected that leads to one of the leaves. The leaves contain a constant value of the objective function, so this approach allows us to obtain a piecewise constant approximation of the objective function in a multidimensional space, see Fig. **4**. Using this approximation, it is possible to estimate the value of the function in those areas where trials have not been carried out, i.e. it is possible to predict the values of the objective function. More details about the decision trees can be found in [33].

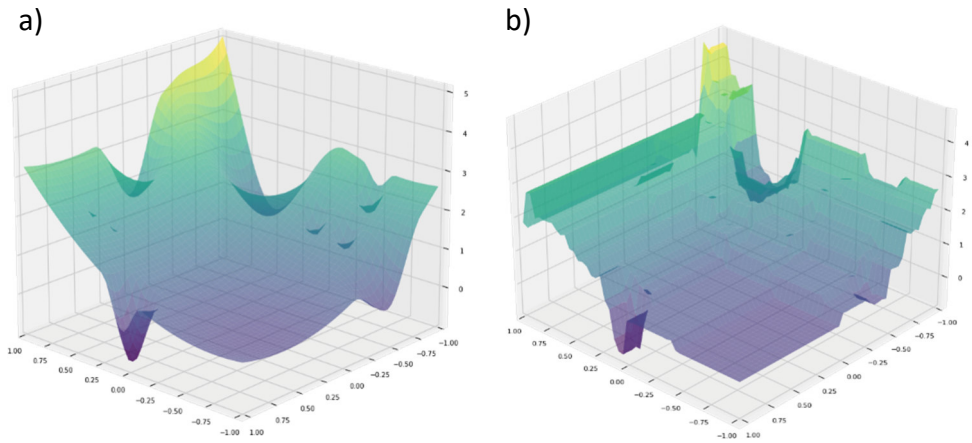

Fig. 4. Example of an objective function from the test class (a), and its piecewise constant approximation (b).

When implementing the described algorithm, we used open source framework of computer vision, image processing, and general-purpose numerical algorithms OpenCV.

3.4 General scheme for using decision trees

At the first stage of the algorithm, a specified number of search trials is carried out using the global search algorithm. At the end of this stage, decision tree based on the accumulated information is created and trained and an approximation of the objective function is constructed. After this, the second stage of the algorithm starts.

When using decision trees for the first time, to simplify the subsequent selection of neighboring points, a uniform grid of a multidimensional search area is constructed with a certain step along each of the coordinates. The values of the objective function approximation are calculated at these points.

Next, for each new trial point, the algorithm evaluates whether this point belongs to the attraction region of the local minimum according to the following algorithm. For a given point, the point closest to the original one in terms of Euclidean distance is determined on the previously constructed uniform grid. The found projection of the original point onto a uniform grid makes it possible to easily identify its neighbors.

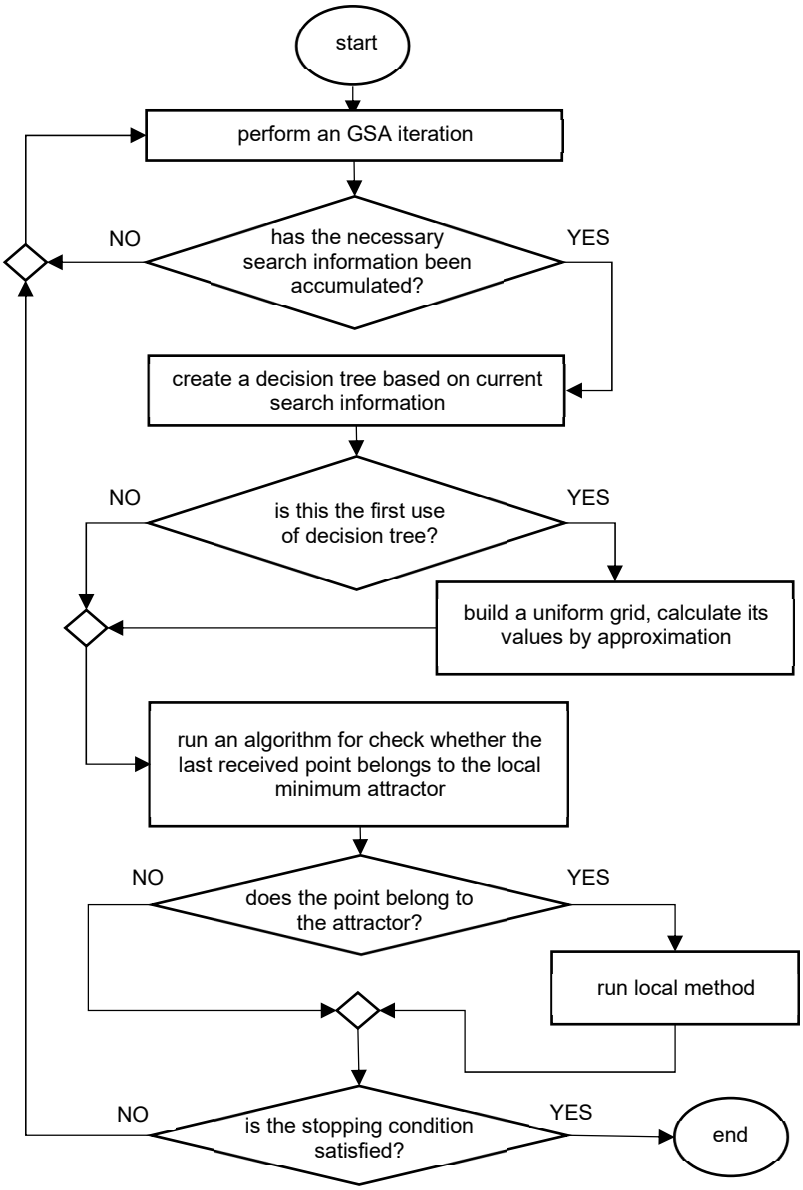

Fig. 5. Flowchart of the algorithm that uses decision tree.

If among the neighbors there are no points at which the function values are less than at the current point, we can assume that we are in the attraction region of the local minimum, otherwise proceed to the end of the iteration. If at least one of the neighboring points has the same function value as the current one, then its neighbors should be checked according to the same scheme. From the area of attraction of the local minimum, a local method is launched, which quickly converges to the local minimum of the function.

4 Results and discussion

4.1 Results of solving a series of test problems by the algorithm using decision trees

Computational experiments were carried out on the Lobachevsky supercomputer. The experiments used the popular benchmark generator for bound-constrained global optimization test functions with known local and global minima GKLS [34], which can generate multiextremal optimization problems with known properties.

Paper [35] provides the results of comparing the global search algorithm under study with the well-known DIRECT [36] and DIRECT*l* [37] methods when solving a series of GKLS problems. The experimental data demonstrate the superiority of GSA over the DIRECT and DIRECT*l* methods in terms of the average number of trials.

Table 1. Averaged number of iterations and speedup of GSA and GSA-DT.

N	Problem class	Averaged number of iterations		Speedup
		GSA	GSA-DT	GSA-DT
2	Simple	2350.0	45.1	11.19
	Hard	4732.3	125.8	6.42
3	Simple	2115.5	206.3	3.64
	Hard	5347.4	355.8	2.54
4	Simple	12168.9	395.7	3.38
	Hard	25636.5	672.1	1.90
5	Simple	20633.6	447.3	8.12
	Hard	126140.4	775.495	6.18

The results of a comparison of the parallel global search algorithm (GSA) and its modification that uses decision tree (GSA-DT) are presenter in Table 1. Parallelization was carried out using MPI technology, 8 processes were used during the experiments. The experiments used a version of the parallel algorithm with synchronous launch of the local method on worker processes. Numerical comparison was carried out on classes of GKLS functions of two types (Simple and Hard) of dimensions 2, 3, 4 and 5.

Parallel GSA-DT has a quite large speedup against the sequential algorithm. The proposed scheme allows one to use the advantages of both approaches: parallelization and fast search for local extrema.

4.2 Tuning hyperparameters using GSA

As mentioned earlier, the hyperparameters of machine learning algorithms are critical to achieving high performance and accuracy of the built models. Hyperparameters, as “regulators” of the model, determine the structure of the model and the way it is trained. Correct parameter values can significantly affect training results, while incorrectly selected hyperparameter values can lead to wrong models.

Each machine learning algorithm has its own characteristics and requires tuning certain parameters to ensure optimal learning and prediction. The effectiveness of parameter tuning using the iOpt framework was tested by solving the following problems:

1. Tuning the parameters of the SVC algorithm (to build a breast cancer prediction model). The regularization coefficient C and the kernel coefficient $gamma$ were considered as tunable hyperparameters.
2. Tuning gradient boosting parameters in XGBoost when building a model on different datasets. Seven parameters were tuned: n_estimators, max_depth, min_child_weight, gamma, subsample, colsample_bytree, learning_rate.

The experiments were carried out on Lobachevsky supercomputer node with two 64-core AMD EPYC 7742 processors (128 cores in total) and 512 GB of RAM.

4.2.1 Tuning a breast cancer prediction model

The study used a well-known dataset to build a breast cancer prediction model [38]. The dataset includes 569 instances, each of which has 30 numerical features. The number of instances in classes is as follows: 212 malignant, 357 benign tumors.

The class prediction model was built using support vector machine (SVC) [39].

The quality metric used was F_1 [40].

Fig. **2** demonstrated the nonlinear dependence of the model quality in F_1 metric on the tunable parameters. In this case, the values of parameters greatly influenced the quality of the prediction.

Table 2. Comparison of iOpt with other frameworks on the classification problem.

Framework	F_1 **score**
----	0.87
Scikit-Optimize	0.90
iOpt	0.975

The problem was solved using scikit-learn with default parameters, as well as after adjusting the parameters using iOpt. With the default parameters, we solved the problem with the value $F_1 = 0{,}87$ (see Table 2). After tuning with iOpt, the quality of model prediction increased.

The use of parallelization allows us to find the parameters for SVC with significant acceleration. The result of tuning the parameters depending on the number of parallel processes used is shown in Table 3.

Table 3. Results of tuning the SVC parameters using iOpt.

Number of processes	F_1 **score**	**Hyperturning time**	**Speedup**
1	0.975	31.199	1
5	0.975	7.876	4.0
10	0.975	4.775	6.5
20	0.975	3.251	9.6
40	0.975	2.002	15.6
80	0.975	1.616	19.3

The lack of linear speedup is due to the relatively short training time of the model for the given parameters C and $gamma$.

4.2.2 XGBoost tuning

XGBoost is one of the implementations of the gradient boosting algorithm based on decision trees. XGBoost has become a staple tool for many data scientists, but using it to improve a model is difficult. This algorithm uses many parameters.

Fig. 6 shows a comparison of the iOpt framework with other well-known frameworks when tuning XGBoost parameters. The experiment was carried out on a series of datasets (BreastCancer, Ecoli, CreditApproval, etc.) using the Optuna [41] and HyperOpt [42] frameworks.

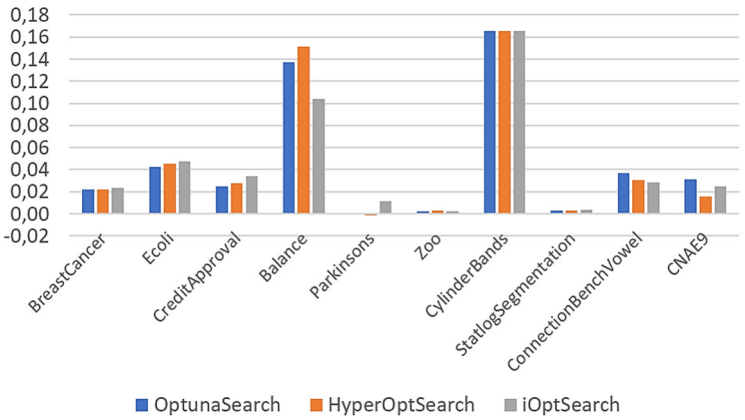

Fig. 6. Comparison of hyperparameter optimization frameworks when tuning XGBoost.

The experiment demonstrates the high efficiency of the iOpt framework along with well-known frameworks for parameters tuning.

5 Conclusion

This paper considers two issues of global optimization and machine learning.

Firstly, global optimization algorithms require new approaches to speed them up due to the growing computational complexity of applied problems. The use of well-known machine learning algorithms allows us to achieve a significant speedup of convergence of global optimization methods.

The experimental results demonstrate the successful integration of the global search algorithm with the local optimization method. Unlike multistart schemes, the decision to launch a local method is made using decision trees. Such combination of methods can significantly speedup the algorithm.

Secondly, global optimization methods themselves help well-known machine learning methods. Since the quality of ML methods is significantly influenced by hyperparameters, and the evaluation of quality metrics is a labor-intensive operation, hyperparameters tuning is a classic area of application for global optimization methods.

Both approaches are implemented in the open-source framework of intelligent optimization methods. The framework is developed in Python and allows one to solve global optimization problems, including tuning the parameters of machine learning methods. The

paper provides a brief description of both approaches, as well as experimental results demonstrating their efficiency and speedup.

Further research may be related to the use of machine learning to speed up algorithms for solving multicriteria optimization problems. In particular, ML methods can be used to approximate and separate the Pareto set from other, non-dominated points in the criteria space. This separation allows us to reduce the number of search trials required to solve the problem. Preliminary results in this direction were obtained in [43] and are the basis for further work.

This work was supported by the Ministry of Science and Higher Education of the Russian Federation, project no. FSWR-2023-0034, and by the Research and Education Mathematical Center "Mathematics for Future Technologies".

References

1. D.R. Jones, M. Schonlau, W.J. Welch, J Glob Optim **13**, 4 (1998)
2. S.A. Piyavskii, Comput. Math. Math. Phys. **12**, 4 (1972)
3. B.O. Shubert, SIAM J Numer Anal **9**, 3 (1972)
4. V.A. Grishagin, R.A. Israfilov, AIP Conf Proc **1738**, 400010 (2016)
5. D. Jones, J. Martins, J Glob Optim **79**, 3 (2021)
6. R. Paulavicius, J. Zilinskas, *Simplicial Global Optimization* (Springer, New York, 2014)
7. R. Paulavicius, Y.D. Sergeyev, D.E. Kvasov, J. Zilinskas, Expert Syst. Appl. **144**, 113052 (2020)
8. Y.D. Sergeyev, D.E. Kvasov, *Deterministic global optimization: An introduction to the diagonal approach* (Springer, New York, 2017)
9. L. Liberti, S. Kucherenko, Int Trans Oper Res **12** (2005)
10. Y.D. Sergeyev, D.E. Kvasov, M.S. Mukhametzhanov, Sci. Rep. **8**, 1 (2018)
11. Y.D. Sergeyev, R.G. Strongin, D. Lera, *Introduction to global optimization exploiting space-filling curves* (Springer Briefs in Optimization, Springer, New York, 2013)
12. R.G. Strongin, Y.D. Sergeyev, *Global optimization with non-convex constraints. Sequential and parallel algorithms* (Kluwer Academic Publishers, Dordrecht, 2000)
13. A. Candelieri, F. Archetti, Soft Comput. **23** (2019)
14. A. Cassioli, D. Di Lorenzo, M. Locatelli et al., Comput Optim Appl **51** (2012)
15. M.K. Bisbo, B. Hammer, Phys. Rev. B **105**, 245404 (2022)
16. S. Zhao, E. Louidor, M. Gupta, Proceedings of the 39th International Conference on Machine Learning, PMLR **162** (2022)
17. S. Wu, Y. Hu, W. Wang, X. Feng, W. Shu, Math. Probl. Eng. **2013** (2013)
18. M. Feurer, F. Hutter, *Hyperparameter Optimization* (Automated Machine Learning, 2019)
19. J. Bergstra, Y. Bengio, J Mach Learn Res **13** (2012)
20. D. von Winterfeldt, E. Ward, *Decision Analysis and Behavioral Research* (1986)
21. A. Klein, S. Falkner, S. Bartels, P. Hennig, F. Hutter, Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, PMLR **54** (2017)
22. J. Wang, S.C. Clark, E. Liu, P.I. Frazier, Oper Res **68** (2016)
23. R. Horst, J Optim Theory Appl **58**, 1 (1988)

24. R. Horst, H. Tuy, J Optim Theory Appl **54**, 2 (1987)
25. V.A. Grishagin, Y.D. Sergeyev, R.G. Strongin, J Glob Optim **10**, 2 (1997)
26. J.D. Pinter, *Global Optimization in Action* (Kluwer Academic Publishers, Dordrecht, 1996)
27. Y.D. Sergeyev, Optimization **44**, 3 (1998)
28. K.A. Barkalov, I.G. Lebedev, Commun. Comput. Inf. Sci. **687** (2016)
29. V.P. Gergel, J Glob Optim **10**, 3 (1997)
30. K.A. Barkalov, Comput. Math. Math. Phys. **42**, 9 (2002)
31. C.T. Kelley, *Iterative Methods for Optimization* (SIAM, Philadelphia, 1999)
32. D. Himmelblau, *Applied Nonlinear Programming* (McGraw-Hill, New York, 1972)
33. S. Brahmbhatt, Practical OpenCV (Apress, New York, 2013)
34. M. Gaviano, D.E. Kvasov, D. Lera, Y.D. Sergeyev, ACM Trans Math Softw **29**, 4 (2003)
35. K.A. Barkalov, V.P. Gergel, J Glob Optim, **66**, 1 (2016)
36. D.R. Jones, *Direct Global Optimization Algorithm* (Encyclopedia of Optimization, 2009)
37. J.M. Gablonsky, C.T. Kelley, J Glob Optim **21**, 1, (2001)
38. Breast Cancer Wisconsin (Diagnostic) Data Set. https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)
39. Chih-Chung Chang, Chih-Jen Lin, ACM Trans Intell Syst Technol **2**, 27 (2011)
40. P.A. Flach, M. Kull, Proceedings of the 28th Advances in Neural Information Processing Systems (2015)
41. T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2019)
42. J. Bergstra, D. Yamins, D.D. Cox, Proceedings of the 12th Python in Science Conference (2013)
43. K.A. Barkalov, V.A. Grishagin, E.A. Kozinov, Lect. Notes Comput. Sci. **13621** (2022)