Automated Parameter Choice with Exploratory Landscape Analysis and Machine Learning

Maxim Pikalov
ITMO University
St. Petersburg, Russia
pikmaksim@gmail.com

Vladimir Mironovich
ITMO University
St. Petersburg, Russia
mironovich.vladimir@gmail.com

ABSTRACT

The performance of an evolutionary algorithm on a particular optimization problem highly depends on the choice of the algorithm's parameters. Fitness landscape analysis can help learn important characteristics of the optimization problem that may influence the optimal parameter values. Using this knowledge for parameter choice can be an important step in designing fitness-aware algorithms.

In this paper, we present an approach to automatic parameter choice that uses exploratory landscape analysis and machine learning. Using a neural network trained on a dataset of optimal parameter choices for particular landscape features of the W-model problem we choose parameters for the $(1+(\lambda,\lambda))$ genetic algorithm. The experimental results suggest that the neural network is capable of providing good parameter choices based on the landscape features computed by the flacco package.

CCS CONCEPTS

• **Theory of computation → Random search heuristics**.

KEYWORDS

Parameter Tuning, Exploratory Landscape Analysis, Black-box Optimization

ACM Reference Format:
Maxim Pikalov and Vladimir Mironovich. 2021. Automated Parameter Choice with Exploratory Landscape Analysis and Machine Learning. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3449726.3463213

1 INTRODUCTION

The development of methods for choosing parameters in evolutionary algorithms is essential for their successful application in practice [11]. There exist several different approaches to this problem, which are commonly referred to as *parameter tuning* and *parameter control*.

Parameter tuning usually involves automated techniques, such as irace [12] or SMAC [6]. These methods are usually based on the evaluation of various parameter combinations on the given problem instance in order to find the best set of parameters. Unfortunately, this may not be possible in a budged-constrained environment, where no time for such training is given.

In parameter control, on the other hand, the parameters are chosen during the optimization process without preliminary training. Thus, this approach can be used more broadly, while also making it possible to change parameters over time as the optimum parameter choice can be quite different for various optimization stages [5, 8].

Fitness landscape analysis [14] is actively studied as a tool that can help extract knowledge from the problem instances, making it possible to tailor the used metaheuristics to the considered optimization problem. *Exploratory landscape analysis* [13] considers fitness landscape features that can be extracted from intermediate solutions and their fitness values, making it possible to gain some understanding of the problem even in the black-box environment.

Exploratory landscape analysis has found various successful applications [15]. One of the promising research directions is automated algorithm selection with exploratory landscape analysis and machine learning [10] and even dynamic algorithm selection [7]. Unfortunately, these works mostly focus on continuous optimization problems and there is still much potential for applying similar ideas in the discrete domain.

In this paper, we present our first steps in developing an approach to automated parameter choice by using fitness landscape analysis together with neural networks. In the proposed approach we evaluate the fitness landscape features of the problem instance and suggest the optimal parameter values by using the neural network trained on a dataset of landscape features and corresponding optimal parameter choices. Compared to automated algorithm selection approaches, we consider the problem of regression of the algorithm's parameters instead of classification of the problem by the best suitable algorithm.

For this study, we collected a dataset of optimal parameter values for the $(1 + (\lambda, \lambda))$ genetic algorithm (GA) [3] with four static parameters on multiple instances of the W-model benchmark problem [17]. We computed the landscape features for the same problem instances using the flacco package in R [9]. Using the neural network trained on this dataset we are able to recommend parameter values for the $(1 + (\lambda, \lambda))$ GA for different optimization problem instances as long as we can compute their fitness landscape features.

The experimental results on various instances of the W-model problem suggest that the proposed landscape-aware approach to parameter choice can help determine good values for static parameters of the $(1 + (\lambda, \lambda))$ genetic algorithm, as the algorithm with suggested parameter values outranks other considered variations of the $(1 + (\lambda, \lambda))$ GA.

GECCO '21 Companion, July 10–14,2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07…$15.00
https://doi.org/10.1145/3449726.3463213

Algorithm 1 The $(1 + (\lambda, \lambda))$ GA with parameters $\lambda_1, \lambda_2, k, c$

$n \leftarrow$ problem size
$\lambda_1 \leftarrow$ mutation phase population size
$\lambda_2 \leftarrow$ crossover phase population size
$k \leftarrow$ mutation coefficient
$c \leftarrow$ crossover probability
Initialize: $x \leftarrow$ uniformly from $\{0,1\}^n$
for $t \leftarrow 1, 2, 3, \ldots$ **do**
  $\ell \sim \mathcal{B}(n, k/n)$
  for $i \in 1, 2, \ldots, \lambda_1$ **do** ▷ Phase 1: Mutation
    $x^{(i)} \leftarrow$ flip ℓ uniformly chosen bits in x
  end for
  $x' \leftarrow$ uniformly from $\{x^{(j)} \mid f(x^{(j)}) = \max\{f(x^{(i)})\}\}$
  for $i \in 1, 2, \ldots, \lambda_2$ **do** ▷ Phase 2: Crossover
    for $j \in 1, 2, \ldots, n$ **do**
      $y_j^{(i)} \leftarrow x'_j$ with probability c, otherwise x_j
    end for
  end for
  $y \leftarrow$ uniformly from $\{y^{(j)} \mid f(y^{(j)}) = \max\{f(y^{(i)})\}\}$
  if $f(y) \geq f(x)$ **then** ▷ Selection
    $x \leftarrow y$
  end if
end for

2 PRELIMINARIES

2.1 $(1 + (\lambda, \lambda))$ Genetic Algorithm

The $(1 + (\lambda, \lambda))$ genetic algorithm [3] is a two-phase genetic algorithm. The algorithm tries to explore the search space by using high mutation rates in the first phase. In the second phase, it applies a crossover mechanism to counteract possible negative effects of high mutation rates on individuals. In this work, we primarily consider the $(1 + (\lambda, \lambda))$ GA with four parameters $\{\lambda_1, \lambda_2, k, c\}$, similar to the one studied in [2]. The algorithm's pseudocode is listed in Algorithm 1 and it works as follows:

- during the first phase of each iteration, the algorithm creates λ_1 mutant individuals by applying the mutation operator with mutation rate k/n;
- the best mutant individual is selected to crossover with the parent individual;
- in the second phase, λ_2 individuals are created with the crossover operator that takes bits from the mutant individual with probability c;
- the best crossover individual replaces the parent if it is at least as good;
- the process repeats until the optimum is found.

For experiments, we use the generic implementation of the $(1 + (\lambda, \lambda))$ genetic algorithm that was first presented in [1].

2.2 W-Model problem

The W-model problem [17] is a configurable benchmark optimization problem. W-model introduces several tunable layers that affect the fitness landscape of the problem and thus make it suitable for various benchmarking tasks. The tunable layers of the W-model problem are:

- *neutrality*, which introduces areas with equal fitness values;
- *epistasis*, which introduces dependency between different individual genes;
- *ruggedness*, which introduces so-called mountains and valleys into the fitness landscape;
- *dummy*, which introduces variables that do not affect the fitness value of the individual.

In this work, we use the C++ implementation of the W-model problem over the OneMax problem on binary strings from the IOHProfiler [4].

2.3 Landscape Analysis

Flacco [9] is a popular R-package for feature-based landscape analysis. It provides various tools for evaluation, analysis and visualization of fitness landscape features.

We use the following feature sets in this study:

- Level-Set: A set of features computed by training classifiers on provided samples of candidate solutions: LDA (Linear Discriminant Analysis), QDA (Quadratic Discriminant Analysis) and MDA (Mixture Discriminant Analysis). Classifiers predict whether or not fitness value falls below the threshold. Statistics of cross-validation mean misclassification errors for each classifier represent the resulting features.
- Meta-Model: A set of features computed by creating linear and quadratic models to approximate the objective function from samples of candidate solutions. Values such as adjusted R^2 of linear and quadratic models, smallest and biggest absolute coefficients of the models and their ratio represent this feature set. These features are designed to approximate the problem structure and complexity with analytical functions and aim to identify the relationship between variables.
- Y-distribution: A set of features computed as the number of peaks, skewness and kurtosis of kernel-based estimation of the density of samples of candidate solutions. Provides an overview of the shape and deeper insight into the landscape and allows to identify areas of high neutrality, ruggedness and points at the presence of plateaus in the search space.

3 EXPERIMENTS

3.1 Performance Dataset

In order to train the neural network, we collected the dataset of average runtimes of the parametrized $(1 + (\lambda, \lambda))$ GA presented in Section 2.1 with different parameter values on different instances of the W-Model problem.

For the $(1 + (\lambda, \lambda))$ GA we consider the following parameters:

- $\lambda_1 \in [2, 3, 4, ..., 9]$;
- $\lambda_2 \in [2, 3, 4, ..., 9]$;
- $k \in [1, 3, 5, 7, 9]$;
- $c \in [0.01, 0.03, 0.05, 0.07, 0.09]$.

We computed the average runtime of $(1 + (\lambda, \lambda))$ GA with all possible combinations of these parameters on the instances of W-Model problem with the following parameters:

- $n \in [8, 16, 32, 64, 128]$;
- dummy $\in [1, 1 - 1/n, 1 - 2/n, ..., 0.8]$;
- neutrality $\in [0, 1, 2, ..., n \times 0.2]$;
- epistasis $\in [0, 1, 2, 3]$.

These values were chosen in order to evaluate problem instances with different landscape features within a reasonable amount of time. In particular, using higher values of epistasis lead to a significant decrease in the performance of some algorithms chosen for comparison. We did not use the ruggedness layer as it does not exactly correlate with the complexity change in the problem.

To sum up, for each combination of $(1 + (\lambda, \lambda))$ GA parameters $\{\lambda_1, \lambda_2, k, c\}$ and W-model problem parameters $\{n$, dummy, neutrality, epistasis$\}$ we run the $(1 + (\lambda, \lambda))$ genetic algorithm 10 times. We choose the set of parameters with the lowest mean runtime as the best set of parameters for the particular problem instance.

3.2 Feature Extraction

In practice, we usually do not know the parameters of the optimized problem, thus we apply the fitness landscape analysis to obtain some insight into the particular problem instance. We computed 10 separate vectors of 35 features with flacco for each W-model problem instance from the performance dataset. This helps us increase the amount of data we can use for the training of the neural network while also taking into the consideration randomness of computed features. Each feature vector is calculated from 100 random individuals which should be taken into account when considering the budget of the algorithm. For now, we do not use the evaluated individuals in the algorithm, although it may be useful to initialize the optimization process from the best individual observed during the evaluation of features.

Combining the features with performance data results in a dataset of best $(1 + (\lambda, \lambda))$ GA parameters for certain fitness landscape features: $features \Rightarrow \{\lambda_1, \lambda_2, k, c\}$. We can then use this data to train the neural network that can suggest certain parameter values for a given set of landscape features.

3.3 Neural Network

We have trained FNN (Feedforward Neural Network) with two hidden layers: 32 and 16 nodes respectively. We chose Adam optimization algorithm as the solver for weight optimization which proved to be the best among SGD (Stochastic Gradient Descent) and Limited-memory BFGS (Broyden-Fletcher-Goldfarb-Shanno algorithm). We used ReLU (Rectified Linear Unit) as an activation function with the regularization parameter set to 10^{-4} while the learning rate was set to constant with the initial value of 10^{-3}. 20% of training data was set aside for validation. The training process was terminated when the validation score did not improve by at least 10^{-4} for 10 consecutive epochs.

3.4 Parameter Recommendation

In order to test the proposed approach to parameter selection we evaluated the performance of the $(1+(\lambda, \lambda))$ genetic algorithm with suggested parameter values on different instances of the W-model problem.

We compare the performance of the algorithm with parameters recommended by the neural network, which we call the $(1 + (\lambda, \lambda))$ GA tuned, with following $(1 + (\lambda, \lambda))$ GA variations available in the generic $(1 + (\lambda, \lambda))$ GA repository:

- $(1 + (\lambda, \lambda)), \lambda = 4$: the $(1 + (\lambda, \lambda))$ GA with default static parameter choices $\lambda_2 = \lambda_1, k = \lambda_1, c = 1/\lambda_1$ for $\lambda_1 = 4$;

Table 1: Comparison of algorithms' performance $n = 256$

Algorithm	Mean rank	Rank SD	Mean diff
(1+1) EA	1.75	1.26	-68.17
RLS	3.02	1.25	-25.45
$(1 + (\lambda, \lambda))$, tuned	3.45	1.46	0
$(1 + (\lambda, \lambda))$, single best	3.81	1.89	84.33
$(1 + (\lambda, \lambda)), \lambda \sim pow(2.5)$	3.99	1.39	59.44
$(1 + (\lambda, \lambda)), \lambda = 4$	6.43	1.15	306.69
$(1 + (\lambda, \lambda)), \lambda \leq n$	6.66	1.44	1949.24
$(1 + (\lambda, \lambda)), \lambda \leq 2lnn$	6.86	1.36	467.93

Table 2: Comparison of algorithms' performance $n = 512$

Algorithm	Mean rank	Rank SD	Mean diff
RLS	2.73	1.83	-184.48
$(1 + (\lambda, \lambda))$, tuned	3.13	1.76	0
$(1 + (\lambda, \lambda)), \lambda \leq n$	3.4	1.91	204.54
$(1 + (\lambda, \lambda)), \lambda \leq 2 \ln n$	3.86	1.18	207.22
$(1 + (\lambda, \lambda)), \lambda = 4$	4.73	0.70	399.1
(1+1) EA	5.33	2.94	886.95
$(1 + (\lambda, \lambda)), \lambda \sim pow(2.5)$	6.13	0.35	706.25
$(1 + (\lambda, \lambda))$, single best	7.66	0.48	1272.10

- $(1 + (\lambda, \lambda)), \lambda \leq 2 \ln n$ and $(1 + (\lambda, \lambda)), \lambda \leq n$: the $(1 + (\lambda, \lambda))$ genetic algorithm with dynamic λ tuned according to the 1/5-th rule with the listed upper bound on λ;
- $(1 + (\lambda, \lambda)), \lambda \sim pow(2.5)$: the $(1 + (\lambda, \lambda))$ genetic algorithm with λ sampled at each iteration from the power-law distribution with $\beta = 2.5$.

We also consider the set of parameters in the training data set, which shows the best overall performance among all parameter choices. The parameter values are $\lambda_1 = 2, \lambda_2 = 2, k = 8, c = 0.02$, and we denote the $(1 + (\lambda, \lambda))$ GA with such parameters as $(1 + (\lambda, \lambda))$, single best. Additionally, we evaluated the (1+1) evolutionary algorithm and random local search (RLS) as baseline for algorithm comparison and to further study the behavior of all considered algorithms on the W-model problem.

For the experimental evaluation, we used the W-model problem instances with size $n = 256$ and $n = 512$. It is important to note that the neural network was trained only on data for $n \leq 128$. The dummy, neutrality and epistasis parameters were chosen from similar ranges as in the performance dataset. We computed 5 separate sets of landscape features for each problem instance. For each set of features, we computed the suggested parameters for the $(1 + (\lambda, \lambda))$ GA using the trained neural network. Finally, we evaluated the performance of all algorithms on these problem instances.

The resulting performance metrics are shown in Table 1 for $n = 256$ and Table 2 for $n = 512$. For each algorithm, we show its mean rank among all the considered algorithms, the standard deviation of the algorithm's rank and the mean difference of the algorithm's runtime in terms of fitness evaluations compared to the tuned $(1 + (\lambda, \lambda))$ GA. Negative difference values indicate that the specific algorithm performs better than our method and positive difference shows the opposite.

Looking into the results we can see that on average the tuned $(1 + (\lambda, \lambda))$ GA is the best-ranked algorithm out of all considered

variations of the $(1 + (\lambda, \lambda))$ GA. In most cases, the performance difference is greater than 100 evaluations required for the evaluation of fitness landscape features. The fitness-aware parameter choice performs better than the $(1+(\lambda, \lambda))$, $\lambda = 4$ with coupled parameters. The tuned $(1 + (\lambda, \lambda))$ GA also outperforms the $(1 + (\lambda, \lambda))$ GA variants with dynamic λ. The dynamic versions of the $(1 + (\lambda, \lambda))$ GA improve their performance with the increased problem size, which may mean that they either need time to find the good parameter values or benefit greatly from changing λ in the different stages of optimization.

The most promising observation is the stark difference of the performance of the tuned $(1 + (\lambda, \lambda))$ GA and single best $(1 + (\lambda, \lambda))$ GA. For $n = 256$ they show similar performance in terms of ranking which may suggest that the neural network just chooses the best set of parameters in the training data. This is proven wrong for $n = 512$, where the single best algorithm shows the worst performance out of all, while the tuned $(1 + (\lambda, \lambda))$ GA still shows great performance. This observation makes us believe that the neural network is capable of learning parameter choices dependent on the landscape features.

Additionally, we examined results for specific W-model parameter ranges in order to better understand the advantages and disadvantages of the trained model and our proposed method. By splitting parameter ranges into intervals, we compared average ranks of different algorithms on the corresponding W-model instances and average differences in the number of function evaluations.

After splitting neutrality and dummy parameter ranges into intervals of 5, we observe that our approach achieves the best results on instances with low or moderate neutrality (≤ 40) with an average rank of about 2.5. Single best $(1 + (\lambda, \lambda))$ GA and power-law $(1 + (\lambda, \lambda))$ GA slightly catch up on intervals with high neutrality (≥ 50) with an average rank of 3.8 and 3.7 respectively, while our solution has the rank of 3.5. Dummy W-model parameter intervals show the opposite results with our solution peaking on moderate or high dummy values (≥ 30) with an average rank of 2.5 while single best and power-law $(1 + (\lambda, \lambda))$ GA show best results on low dummy instances (≤ 20) with average ranks of 3.8 and 3.9 respectively, while our solution has an average rank of 3.4.

4 CONCLUSION AND FUTURE WORK

Successful application of exploratory landscape analysis and machine learning techniques in the continuous optimization domain motivates similar research in the discrete domain. With this work, we make the first step towards automated parameter choice using the fitness landscape analysis and neural networks for discrete optimization problems.

In this study, we collected a dataset of optimum parameter choices for the $(1 + (\lambda, \lambda))$ GA on multiple instances of the W-model benchmark problem. Using a neural network trained on this dataset we choose four parameters in the $(1 + (\lambda, \lambda))$ algorithm tailored to the considered optimization problem instance. The experimental results show that the proposed landscape-aware approach to the parameter choice can help determine good values for static parameters of the $(1 + (\lambda, \lambda))$ genetic algorithm.

For future work, it is important to further evaluate the proposed approach on different optimization problems, as well as other algorithms. Examining problems with features not represented in the W-model can help improve the training data, while successfully applying the proposed approach to other problems with similar characteristics would further show the robustness of our method.

Finally, our primary goal is the development of a dynamic parameter tuning approach that would use the individuals evaluated by the optimization algorithm for the computation of landscape features on the fly. It is important to consider how the use of such individuals may affect the performance of our approach, as landscape features were shown to be sensitive to the sampling strategy [16].

ACKNOWLEDGMENTS

The reported study was funded by RFBR and CNRS, project number 20-51-15009.

REFERENCES

[1] Anton Bassin and Maxim Buzdalov. 2020. The $(1 + (\lambda, \lambda))$ Genetic Algorithm for Permutations. In *Proceedings of Genetic and Evolutionary Computation Conference Companion*. ACM, 1669–1677.

[2] Nguyen Dang and Carola Doerr. 2019. Hyper-Parameter Tuning for the $(1 + (\lambda, \lambda))$ GA. In *Proceedings of Genetic and Evolutionary Computation Conference*. 889–897.

[3] Benjamin Doerr, Carola Doerr, and Franziska Ebel. 2015. From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science* 567 (2015), 87–104.

[4] Carola Doerr, Hao Wang, Furong Ye, Sander van Rijn, and Thomas Bäck. 2018. *IOHprofiler: A Benchmarking and Profiling Tool for Iterative Optimization Heuristics*. https://arxiv.org/abs/1810.05281 IOHprofiler is available at https://github.com/IOHprofiler.

[5] Ágoston Endre Eiben, Zbigniew Michalewicz, Marc Schoenauer, and J. E. Smith. 2007. Parameter control in evolutionary algorithms. In *Parameter Setting in Evolutionary Algorithms*. Number 54 in Studies in Computational Intelligence. Springer, 19–46.

[6] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *Proceedings of Learning and Intelligent Optimization*. Springer, 507–523.

[7] Anja Janković and Carola Doerr. 2019. Adaptive landscape analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 2032–2035.

[8] Giorgos Karafotias, Mark. Hoogendoorn, and Ágoston E. Eiben. 2015. Parameter Control in Evolutionary Algorithms: Trends and Challenges. *IEEE Transactions on Evolutionary Computation* 19, 2 (2015), 167–187.

[9] Pascal Kerschke and Heike Trautmann. 2016. The R-package FLACCO for exploratory landscape analysis with applications to multi-objective optimization problems. In *2016 IEEE Congress on Evolutionary Computation*. IEEE, 5262–5269.

[10] Pascal Kerschke and Heike Trautmann. 2019. Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. *Evolutionary computation* 27, 1 (2019), 99–127.

[11] Fernando G. Lobo, Cláudio F. Lima, and Zbigniew Michalewicz (Eds.). 2007. *Parameter Setting in Evolutionary Algorithms*. Number 54 in Studies in Computational Intelligence. Springer.

[12] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari. 2016. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives* 3 (2016), 43–58.

[13] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. 2011. Exploratory landscape analysis. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. 829–836.

[14] Olaf Mersmann, Mike Preuss, and Heike Trautmann. 2010. Benchmarking evolutionary algorithms: Towards exploratory landscape analysis. In *International Conference on Parallel Problem Solving from Nature*. Springer, 73–82.

[15] Gabriela Ochoa and Katherine Malan. 2019. Recent Advances in Fitness Landscape Analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Prague, Czech Republic) *(GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 1077–1094. https://doi.org/10.1145/3319619.3323383

[16] Quentin Renau, Carola Doerr, Johann Dreo, and Benjamin Doerr. 2020. Exploratory landscape analysis is strongly sensitive to the sampling strategy. In *International Conference on Parallel Problem Solving from Nature*. Springer, 139–153.

[17] Thomas Weise and Zijun Wu. 2018. Difficult Features of Combinatorial Optimization Problems and the Tunable W-Model Benchmark Problem for Simulating them. In *Proceedings of Genetic and Evolutionary Computation Conference Companion*. 1769–1776.