REPORT

Research Project

Investigating machine learning algorithms for adapting meta-heuristics parameters

KANAN MIKAYILOV
RUFAT HUSEYNOV

MASTER OF SCIENCE IN DATA SCIENCE AND ARTIFICIAL INTELLIGENCE

FRENCH-AZERBAIJANI UNIVERSITY

9TH JUNE 2022

Contents

1 Introduction

Meta-heuristics are powerful stochastic optimization algorithms that showed a high success when solving a wide range of optimization problems. However, an offline tuning of their parameters may fail given the different characteristics of optimization problems. Therefore, a parameter adaptation that considers feedback from the search process can be an alternative choice. More precisely, archive-based strategies are adopted, where successful parameters from the search history are stored and used in the next iterations for generating potentially better solutions.

2 Objective

The main objective of this research project is to exploit machine learning capabilities (regression, classification and / or clustering) for generating an appropriate set of parameters for each iteration of the optimization process.

The first part of the research project was dedicated to a review of the state-of-the-art on parameter adaptation strategies for meta-heuristics. The second part of the research project was the proposition of a novel strategy. The third part concerned the implementation and experimenting the proposal on a recent set of optimization problems.

3 Optimization

Optimization is the process of finding the most optimal value from a set of values, with regard to some criterion. It can be the either maximizing or minimizing the output of a real function. Currently, the optimization area is very demanded, and the researchers are being interested in for centuries.

As it is shown in Figure 1, all functions have minimal and maximum value, whether globally or locally. Optimization area is concentrated exactly on finding those critical values.

There are a lot of algorithm categories that are implemented for the optimization problems, such as **Direct algorithms**, **Stochastic algorithms**, or **Population**

algorithms. In this project the result is obtained by the help of population-based algorithms.

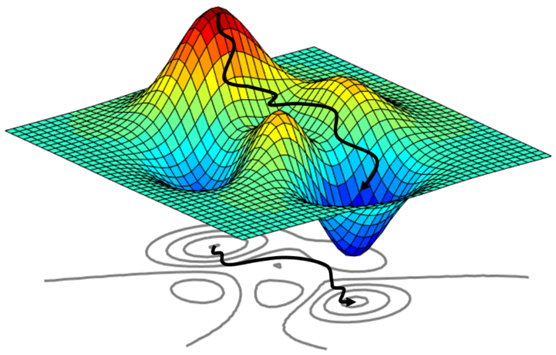


Figure 1: Optimization of a function

3.1 Population algorithms

Population algorithm is an optimization algorithm, that uses possible solutions, which together form a population, to find the most optimal solution for the problem. Each possible solution is called an individual, and is usually generated randomly inside the boundaries. The algorithms such as **Genetic algorithm**, **Differential Evolution algorithm**, and **Particle Swarm Optimization algorithm** all belong to population-based algorithms.

The earlier mentioned algorithms are also **meta-heuristic algorithms**, and it is very crucial to research this therm before going to implementation.

3.2 Meta-heuristics

A meta-heuristics is a procedure designed to generate a search algorithm that may provide a sufficiently good solution to an optimization problem, Meta-heuristics do not always find globally optimal solution. By searching over a large set of solutions (individuals inside the population), meta-heuristics can often find good solutions with little computational effort.

4 Differential Evolution Algorithm

This project is mainly based on DE algorithm and its variants. It is a Evolutionary algorithm (EA), which is based on natural selection, those algorithms mainly contain following stages, **initialization**, **selection**, **mutation**, **crossover**, **evaluation**, for each generation.

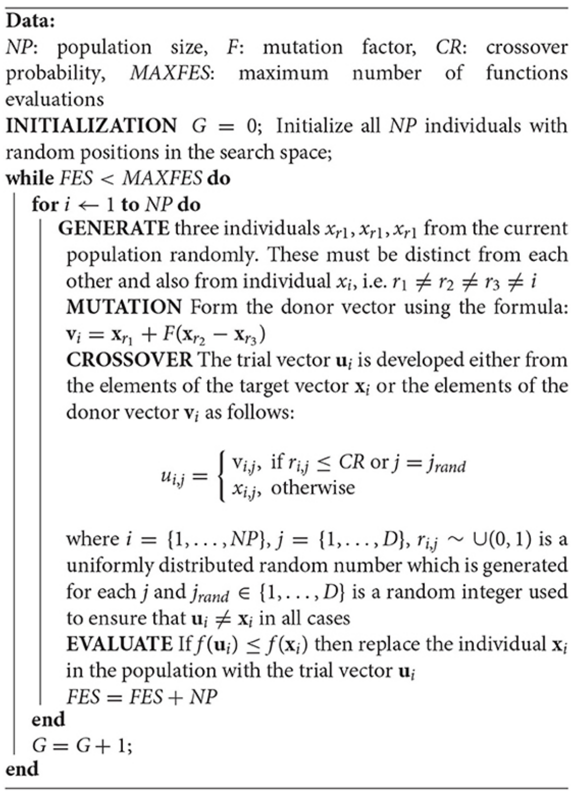
The main difference between classical optimization techniques and evolutionary algorithms is that EA do not make sure about finding the optimum parameter values for a problem. On the other hand, EA are flexible when solving different types of problems. EA tries to evolve an individual into one with a better fitness value through a “trial individual”. To generate a trial individual, existing individuals are chosen as raw genetic material and combined using various genetic operators. If the trial individual has a better fitness value than the original individual, the trial individual replaces it in the next generation population. EA-s radically differ from one another based on their strategies for generating trial individuals.

The algorithms search the design space by maintaining a population of candidate solutions (individuals) and creating new solutions by combining existing ones according to a specific process. The candidates with the best objective values are kept in the next iteration of the algorithm in a manner that the new objective value of an individual is improved forming consequently part of the population, otherwise the new objective value is discarded. The process repeats itself until a given termination criterion is satisfied.

Differential Evolution Algorithm has also its variants, such as composite DE (CODE), self-adaptive DE (JDE), that are created to find the optimal value, with a better convergence rate, which means, in less time and complexity.

4.1 Standard Differential Evolution - SDE

This algorithm is a main DE algorithm, and it has 2 parameters that should be passed to, such as F - mutation factor, CR - crossover probability. Both these parameters are used to generate new trial individual for each iteration, checking if it is better than current individual, and updating the population, in case if the trial vector is better. This algorithm is better than other variants in case of convergence rate, which means that it takes this algorithm less iterations to find optimal value than other variants.



```
Data:
NP: population size, F: mutation factor, CR: crossover probability, MAXFES: maximum number of functions evaluations
INITIALIZATION $G = 0$; Initialize all NP individuals with random positions in the search space;
while $FES < MAXFES$ do
    for $i \leftarrow 1$ to NP do
        GENERATE three individuals $x_{r1}, x_{r1}, x_{r1}$ from the current population randomly. These must be distinct from each other and also from individual $x_i$, i.e. $r_1 \neq r_2 \neq r_3 \neq i$
        MUTATION Form the donor vector using the formula: $\mathbf{v}_i = \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3})$
        CROSSOVER The trial vector $\mathbf{u}_i$ is developed either from the elements of the target vector $\mathbf{x}_i$ or the elements of the donor vector $\mathbf{v}_i$ as follows:
            $u_{i,j} = \begin{cases} v_{i,j}, & \text{if } r_{i,j} \leq CR \text{ or } j = j_{rand} \\ x_{i,j}, & \text{otherwise} \end{cases}$
        where $i = \{1, \ldots, NP\}$, $j = \{1, \ldots, D\}$, $r_{i,j} \sim \cup(0,1)$ is a uniformly distributed random number which is generated for each $j$ and $j_{rand} \in \{1, \ldots, D\}$ is a random integer used to ensure that $\mathbf{u}_i \neq \mathbf{x}_i$ in all cases
        EVALUATE If $f(\mathbf{u}_i) \leq f(\mathbf{x}_i)$ then replace the individual $\mathbf{x}_i$ in the population with the trial vector $\mathbf{u}_i$
        $FES = FES + NP$
    end
    $G = G + 1$;
end
```

Figure 2: Pseudo-code of standard DE

4.2 Composite Differential Evolution - CODE

For composite DE algorithm, it is used to pass 3 pairs of parameters, also called parameter pool, as during the algorithm execution, it takes randomly selected parameter pair to generate the trial individual. Also different DE algorithms use different generation schemes to generate the trial individual, such as **DE/rand/1**, which is the basic scheme, **DE/best/1**, **DE/current-to-best/1**, **DE/current-to-rand/1**, etc. For example, this algorithm uses **rand/1/bin**, **rand/2/bin**, **current-to-rand-1** schemes, and for each scheme it uses randomly selected parameter pair from the parameter pool. This algorithm takes a lot more time to be executed than standard DE, but it has a better convergence rate.

```
Data: $NP$, $MAXFES$
INITIALIZATION $G = 0$; $FES = NP$;
while $FES < MAXFES$ do
    $P_{G+1} = 0$
    for $i \leftarrow 1$ to $NP$ do
        GENERATE three trial vectors $u_{i_1,G}, u_{i_2,G}$ and $u_{i_3,G}$ for
        the target vector $x_{i,G}$ using the three generation schemes
        "rand/1/bin," "rand/2/bin," and "current-to-rand/1," each
        with control parameter setting randomly selected from
        the parameter candidate pool: $[F = 1.0, CR = 0.1]$,
        $[F = 1.0, CR = 0.9]$ and $[F = 0.8, CR = 0.2]$.
        EVALUATE the objective function value of the three trial
        vectors $u_{i_1,G}, u_{i_2,G}$, and $u_{i_3,G}$
        CHOOSE the best trial vector $u^*_{i,G}$ from the three trial
        vectors
        $P_{G+1} = P_{G+1} Uselect(x^*_{i,G}, u^*_{i,G})$
        $FES = FES + 3$
    end
    $G = G + 1$;
end
```

Figure 3: Pseudo-code of composite DE

4.3 Self-adaptive Differential Evolution - JDE

This algorithm uses the same generation scheme as standard DE, but in standard DE algorithm, the parameters are kept fixed throughout the execution. Therefore, self-adaptive DE generated new parameters based on a criterion in each generation, which leads to better results, than standard DE and composite DE in regard of convergence rate.

$$F_{i,G+1} = \begin{cases} F_l + rand_1 F_u, & \text{if } rand_2 < \tau_1 \\ F_{i,G}, & \text{otherwise} \end{cases}$$

$$CR_{i,G+1} = \begin{cases} rand_3, & \text{if } rand_4 < \tau_2 \\ CR_{i,G}, & \text{otherwise} \end{cases}$$

Figure 4: self-adaptive DE parameter generation criterion

5 Parameter tuning

DE algorithms has an access to many parameters: number of iterations, population size, dimension size, maximum function evaluation, upper boundary, lower boundary, but there are 2 crucial parameters that can have a great impact on finding the most optimal value, such as mutation factor - F and crossover probability - CR. Before, all DE algorithms used either fixed parameters, or randomly generated parameters in each iteration. The latter is a bit better than the former case, but it also is not the most optimal way to implement meta-heuristics.That is why, we introduce Machine Learning (ML) techniques to tune the parameters in each generation.

5.1 Machine Learning

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence (AI) based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. With ML, it's possible to quickly and automatically produce models that can analyze bigger, more complex data and deliver faster, more accurate results – even on a very large scale.

ML has 2 categories: supervised and unsupervised. Supervised ML techniques are applied when we have a piece of data that we want to predict or explain. Unsupervised ML looks at ways to relate and group data points without the use of a target variable to predict. There are ML techniques, such as Regression, Classification, Clustering.

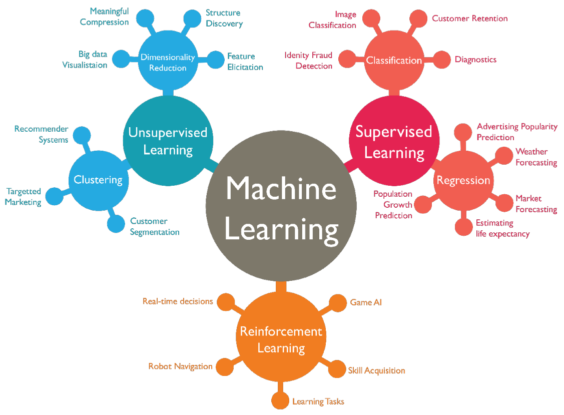


Figure 5: Machine learning

Which techniques are used for prediction in this project?

F and CR are used in each generation, that means for each individual, so at the beginning of each generation, ML techniques should be called to predict, if a randomly generated value is promising for the current population or not.

The prediction stage starts from the iteration 11, and before the iteration 11, our mission is to collect the data.

After generation 11, each time after generating the parameters randomly, they are being tested, if they will be promising or not, by using the techniques to predict. For this mission, the linear regression technique is used, as it is the simplest technique that is very helpful for such a simple usage. First, Linear Regression should train on some data, which is collected from generation 1 to 10. Then, is the parameter is checked as not promising, the parameter is randomly regenerated, until finding the possible promising parameter value.

How the data is collected?

Before the generation 11, at the beginning of each generation, parameters are generated randomly inside their boundaries, and at the end of the generation, the individual is updated, if trial solution is better than current solution. In that case, the current parameters are set as promising, otherwise as not promising.

6 Population reduction

The usage of ML techniques is already made DE algorithm better by means of convergence rate. Now, the parameters that can help find optimal solution are not fixed, and also not randomly generated, they are predicted, and only good parameters are taken, so it is expected that for each generation, the probability of creation of a trial solution that is better than current solution is really high, hence, the high probability of getting more promising solutions.

If it is possible to get the optimal value from better solutions, then why not remove all worst solutions from the population at the end of each generation, so that optimal solution can be found in less time.

The algorithm is called **Linear Population Size Reduction**, and it does not use any additional parameter, but only parameters that is already used by DE algorithm: initial population size, minimum population size, current population size, current generation, maximum generation.

$$N_{G+1} = \text{round}\left[\left(\frac{N^{min} - N^{init}}{MAX\_NFE}\right) \cdot NFE + N^{init}\right]$$

Figure 6: Population reduction formula

7 Benchmark functions

To test the implemented algorithms, several benchmark functions were used.

7.1 Ackley

$$f(\mathbf{x}) = -a \exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

Figure 7: Ackley function

7.2 Rastrigin

$$f(\mathbf{x}) = 10d + \sum_{i=1}^{d}\left[x_i^2 - 10\cos(2\pi x_i)\right]$$

Figure 8: Rastrigin function

7.3 Rosenbrock

$$f(\mathbf{x}) = \sum_{i=1}^{d-1}\left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$$

Figure 9: Rosenbrock function

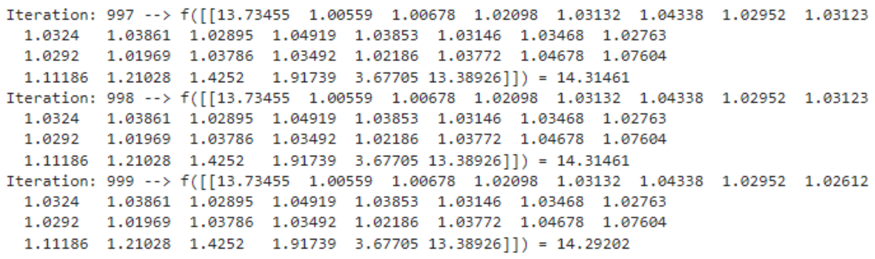
7.4 Schwefel

$$f(\mathbf{x}) = 418.9829d - \sum_{i=1}^{d} x_i \sin(\sqrt{|x_i|})$$

Figure 10: Schwefel function

8 Results

As a result, for each algorithm implemented, the best solution in the population is shown each generation, the graph of best values is shown at the end of execution, and the time it took to execute the algorithm.



```
Iteration: 997 --> f([[13.73455  1.00559  1.00678  1.02098  1.03132  1.04338  1.02952  1.03123
  1.0324   1.03861  1.02895  1.04919  1.03853  1.03146  1.03468  1.02763
  1.0292   1.01969  1.03786  1.03492  1.02186  1.03772  1.04678  1.07604
  1.11186  1.21028  1.4252   1.91739  3.67705 13.38926]]) = 14.31461
Iteration: 998 --> f([[13.73455  1.00559  1.00678  1.02098  1.03132  1.04338  1.02952  1.03123
  1.0324   1.03861  1.02895  1.04919  1.03853  1.03146  1.03468  1.02763
  1.0292   1.01969  1.03786  1.03492  1.02186  1.03772  1.04678  1.07604
  1.11186  1.21028  1.4252   1.91739  3.67705 13.38926]]) = 14.31461
Iteration: 999 --> f([[13.73455  1.00559  1.00678  1.02098  1.03132  1.04338  1.02952  1.02612
  1.0324   1.03861  1.02895  1.04919  1.03853  1.03146  1.03468  1.02763
  1.0292   1.01969  1.03786  1.03492  1.02186  1.03772  1.04678  1.07604
  1.11186  1.21028  1.4252   1.91739  3.67705 13.38926]]) = 14.29202
```

Figure 11: Generation best solution

8.1 Standard DE

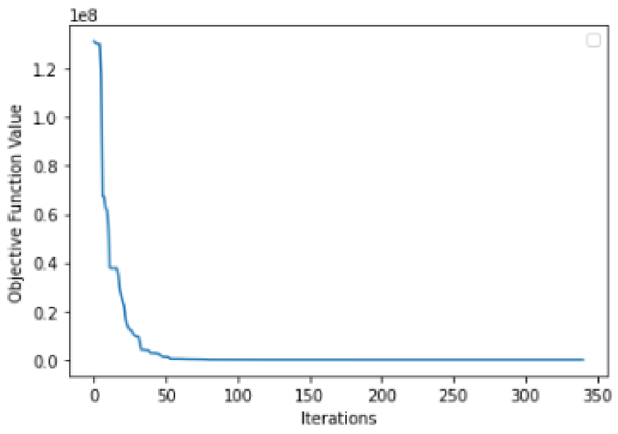


Figure 12: Standard DE graph

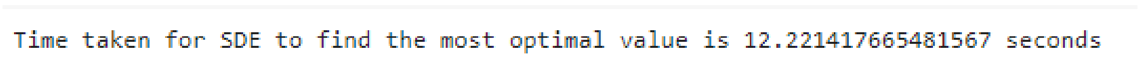

```
Time taken for SDE to find the most optimal value is 12.221417665481567 seconds
```

Figure 13: Standard DE execution time

8.2 Standard DE with ML techniques

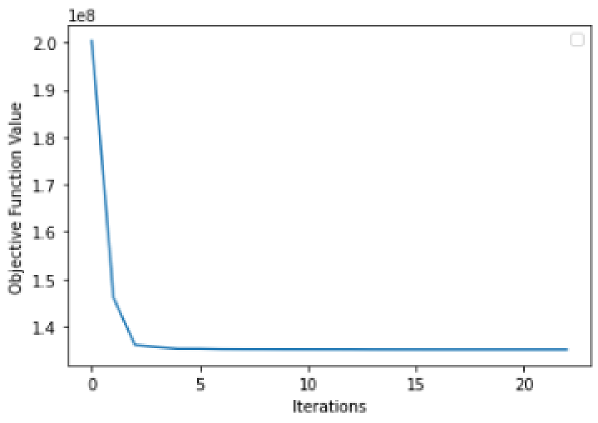


Figure 14: Standard DE with ML graph

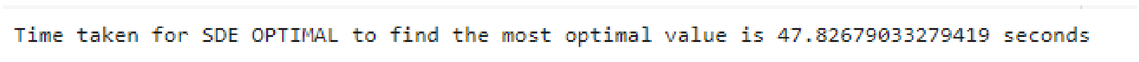

```
Time taken for SDE OPTIMAL to find the most optimal value is 47.82679033279419 seconds
```

Figure 15: Standard DE with ML execution time

9 Conclusion

As a conclusion, in this research project, there were implemented many meta-heuristic optimization algorithms, such as DE with its variants CODE, JDE, JADE, SADE, also SDE with Machine Leaning involved, which made it even better than former variants.

As a result, those above mentioned algorithms were compared by means of time complexity, and convergence rate (optimal value stability).

For time complexity, it is concluded that the algorithms go in the following order (from fastest to slowest):

$$SDE \rightarrow JDE \rightarrow JADE \rightarrow SADE \rightarrow SDE\_OPTIMAL \rightarrow CODE$$

For convergence rate, it is concluded that the algorithms go in the following order (from best to worst):

$$SDE\_OPTIMAL \rightarrow CODE \rightarrow JDE \rightarrow SDE \rightarrow JADE \rightarrow SADE$$

10 Reference

Rawaa Dawoud Al-Dabbagh, Ferrante Neri, Norisma Idris, and Mohd Sapiyan Baba. Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy. Swarm and Evolutionary Computation, 43:284–311, 2018.

Mokhtar Essaid, Mathieu Brévilliers, Julien Lepagnot, Lhassane Idoumghar, and Daniel Fodorean. An eigenvector-enhanced parallel adaptive differential evolution for electric motor design. In 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), pages 713–720, 2019.

Mokhtar Essaid, Mathieu Brévilliers, Julien Lepagnot, Lhassane Idoumghar, and Daniel Fodorean. Hybrid parameter adaptation strategy for differential evolution to solve real-world problems. In 2019 IEEE Congress on Evolutionary Computation (CEC), pages 3030–3036, 2019.

Claudia V. Pop, Mokhtar Essaid, Lhassane Idoumghar, and Daniel Fodorean. Novel differential evolutionary optimization approach for an integrated motor-magnetic gear used for propulsion systems. IEEE Access, 9:142114–142128, 2021.

Georgioudakis, M. and Plevris, V., 2022. A Comparative Study of Differential Evolution Variants in Constrained Structural Optimization.

Ryoji Tanabe and Alex S. Fukunaga, Graduate School of Arts and Sciences The University of Tokyo, Improving the Search Performance of SHADE Using Linear Population Size Reduction,