REPORT

Backtracking Search Optimization Algorithm

Rufat Huseynov
rufat.huseynov@ufaz.az
Kanan Mikayilov
kanan.mikayilov@ufaz.az

Master of Science in Data Science and Artificial Intelligence
French-Azerbaijani University
18.12.2021

Contents

1 Introduction

Backtracking Search Optimization Algorithm (BSA) is one the most widely used algorithms used to solve complex optimization problems. BSA is a type of Evolutionary Algorithm (EA) with a metaheuristic approach to solving problems that cannot be solved with regularly used algorithms. The goal of EAs is to find near-optimal solutions to the given problems. The main difference between classical optimization techniques and evolutionary algorithms is that EAs do not make sure about finding the optimum parameter values for a problem. On the other hand, EAs 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.

2 Backtracking Search Optimization Algorithm

2.1 General Description

BSA - a new type of Evolutionary Algorithm has its unique mechanism used to generate a trial individual makes it possible to solve numerical optimization problems easier and quickly. BSA uses three basic genetic operators - selection, mutation and crossover - to generate trial individuals. BSA has a random mutation strategy that uses only one genetic algorithms such as DE and its derivatives JDE, JADE and SADE. BSA randomly chooses the direction individual from individuals of a randomly chosen previous generation. BSA uses a non-uniform crossover strategy that is more complex than the crossover strategies used in many genetic algorithms. The structure of the algorithm is following:

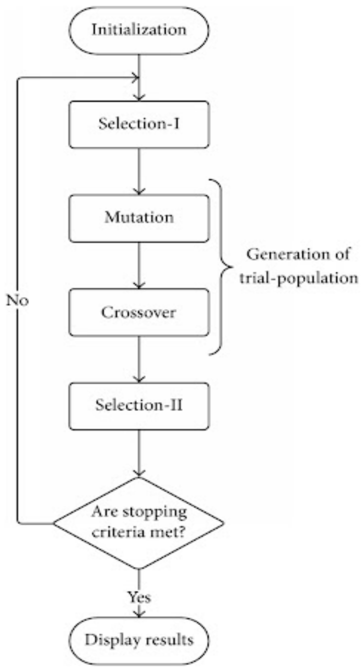


Figure 1: The structure of BSA

2.2 Mechanisms

Backtracking Search Optimization Algorithm consists of 5 evolutionary mechanisms:

1. Initialization
2. Selection-I
3. Mutation (Generation of trial-population)
4. Crossover (Generation of trial-population)
5. Selection-II

Let's look at them in a more detailed way.

Initialization

This step is used to initialize the main data, crucial in calculation of minimum value, which is following.

$$P_{i,j} = U(0,1) \times (up_j - low_j) + low_j \tag{1}$$

With Eq. (1), Population is set, the main feature of this algorithm, as it is already been told, BSA is a population-based algorithm, which means that to find the optimal value for functions the algorithm works with matrices, so called populations. Also, old population is set by using this equation, which is crucial to change the search direction for minimum value in the beginning of each epoch.

For i=1,2,3,..,N (population size), j=1,2,3,..,D (solution dimension), U = uniform distribution.

$$fitP_i = fnc(P_i) \tag{2}$$

In Eq. (2), there is fitness population array, is used to find keep the values calculated by the function to give the minimum and optimal value at the end of the algorithm. Here, fnc() is the function chosen by the user, from following: Ackley, Rastrigin, Rosenbrock, Schwefel.

Selection-I

This stage is the first stage of the loop. The main objective of this stage is to update the old population, to change the search direction of the algorithm, by the following equations:

$$if \quad U(0,1) < U(0,1) \quad then \quad oldP_{i,j} = P_{i,j}$$

$$oldP = permute(oldP)$$

where permute() is the random shuffling function and by using this function oldP matrix is being shuffled (rows) at the beginning of each epoch.

Mutation

$$Mutant = P + F \times (oldP - P) \tag{3}$$

In Eq (3), Mutant is matrix, which will be used to generate the trial population - the second main attribute of the algorithm. Here, F is the variable which is used to control amplitude of search direction of our matrix, which is the difference between old population and current population, and is calculated, by using the following equation:

$$F = 3 \times N(0,1)$$

where N() is standard normal distribution.

Crossover

This stage is used to update the trial population to its final form before evaluation, which is generated from the mutant matrix. This stage is divided into 2 parts. First part is to generate matrix of ones and zeros of size NxD, which will be used to control the trial population.

```
$map_{(1:N,1:D)}$=1   // Initial-map is an N-by-D matrix of ones.
if $a < b \mid a, b \sim U(0,1)$ then
    for $i$ from 1 to $N$ do
        $map_{i,u_{(1:\lceil mixrate \cdot rnd \cdot D \rceil)}} = 0 \mid u = permuting(\langle 1, 2, 3, ..., D \rangle)$
    end
else
    for $i$ from 1 to $N$ do, $map_{i,randi(D)} = 0$, end
end
$T := Mutant$   // Initial T
for $i$ from 1 to $N$ do
    for $j$ from 1 to $D$ do
        if $map_{i,j} = 1$ then $T_{i,j} := P_{i,j}$
    end
end
```

Here, u is an array of permuted values from 1 to D (solution dimension), := is update (assignment) operation, randi = U(0,D), $\lceil x \rceil$ is ceiling function

The second part is called Boudary Control mechanism and used to update the trial population based on the values received from the first part, by checking the values to be inside the given boundaries (lower, upper).

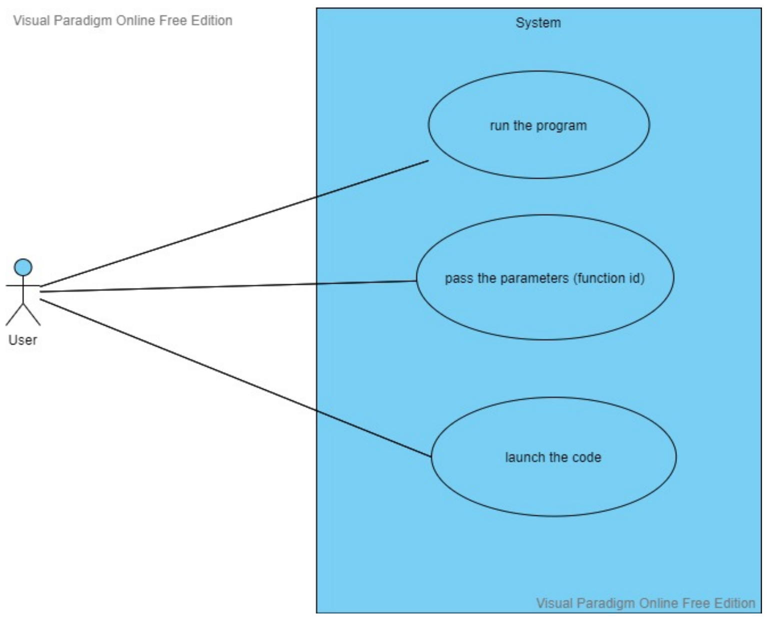
$$\begin{aligned} &if \quad T_{i,j} < low_j \quad or \quad T_{i,j} > up_j \quad then \\ &\quad T_{i,j} = U(0,1) \times (up_j - low_j) + low_j \end{aligned}$$

Selection-II

The objective of this stage is to update the fitness values, and to update the global minimum and minimizer. In initialization the fitnessP array was defined and the values calculated by passing the values of population to chosen function were assigned to fitnessP array. At this stage fitnessT array is defined similarly to the stage initialization, but in this case we pass T array to the function during each epoch. Thus, the algorithm gets the updated values from the functions, which could be better (less), so by comparing 2 fitness arrays (fitP, fitT), fitP is updated, if there is any less value in fitT. Also in case fitT has better value, global minimizer is also updated. This stage is based on a greedy selection.

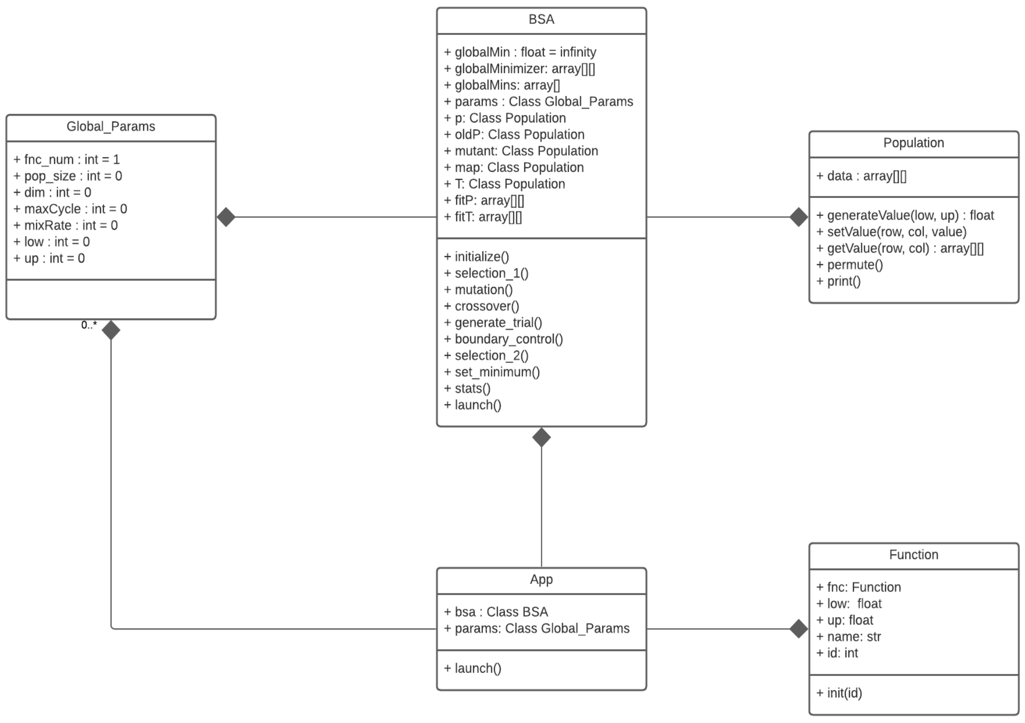
3 UML Diagrams

3.1 Use-Case Diagram



Visual Paradigm Online Free Edition
System
run the program
pass the parameters (function id)
launch the code
User
Visual Paradigm Online Free Edition

3.2 Class Diagram



4 Implementation

To implement BSA algorithm, Python programming language was used including its libraries. There were other programming languages such as R, MATLAB, Julia that we could use to develop BSA algorithm. However, we came into account that Python would suit best for this mission, especially its library called NumPy is capable of doing most of the required calculation. NumPy is quite useful in Data Handling processes. Below you can view classes and their functionalities used in this project.

4.1 Methods

class Population:

- generateValue(low, up) - Function generates value by using Eq (1)
- setValue(row,col,value) - function gets value and assigns it to population that called this function on [row][col] cell
- getValue(row, col) - returns the value on [row][col] cell of the population
- permute() - function shuffles the rows of the population
- print() - function that beautifully prints the population matrix

class BSA:

Next methods implement stages from mechanism section above:

- initialize()
- selection_1()
- mutation()
- crossover()
- generate_trial()
- boundary_control()
- set_minimum()

- selection_2()

stat() method prints mean, best, worst, standard deviation and variance of the array consisting the global minimums recorded at the end of epochs after 30 runs.

launch() method implements the structure of the BSA algorithm shown in Figure 1 above.

class App:

- launch() - method takes the function id from user and implements makes an instance of BSA class to be launched later

Benchmark functions

Ackley

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

Rastrigin

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

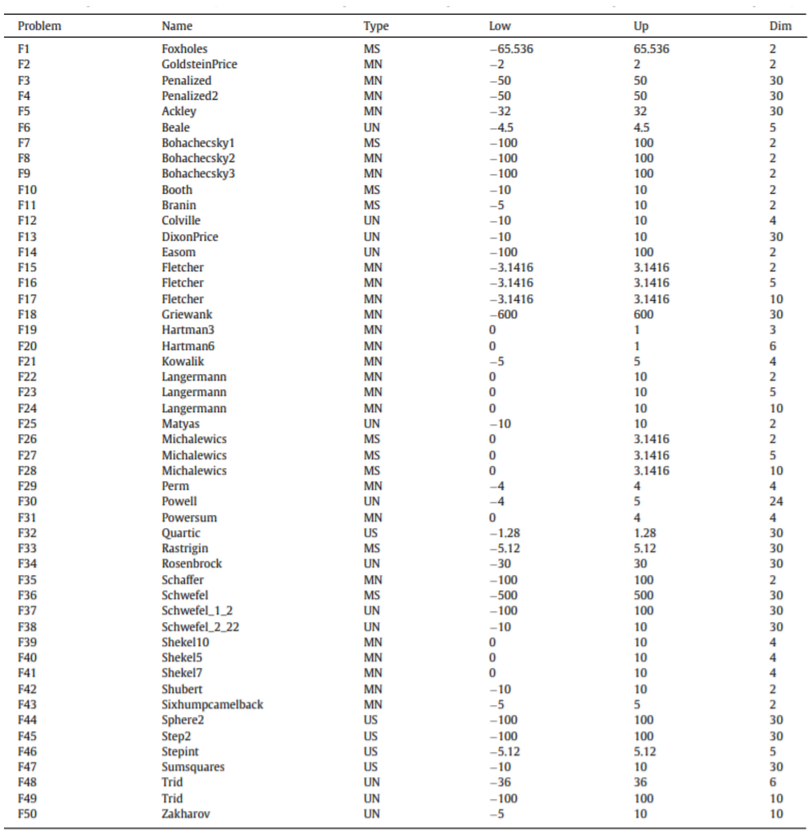
Rosenbrock

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

Schwefel

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

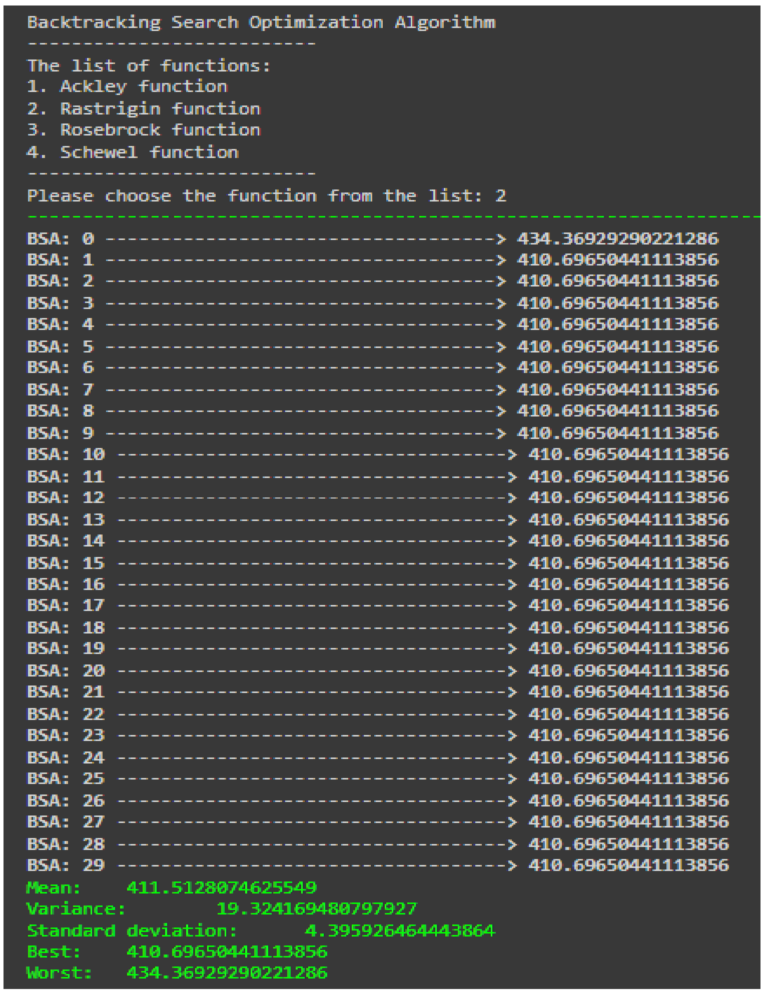
The lower and upper bounds, also dimension size for each function are taken from the figure below:



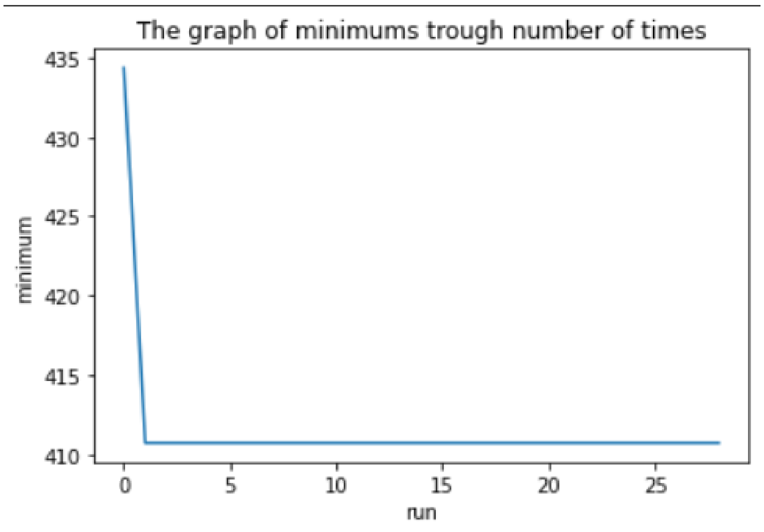
Problem	Name	Type	Low	Up	Dim
F1	Foxholes	MS	−65.536	65.536	2
F2	GoldsteinPrice	MN	−2	2	2
F3	Penalized	MN	−50	50	30
F4	Penalized2	MN	−50	50	30
F5	Ackley	MN	−32	32	30
F6	Beale	UN	−4.5	4.5	5
F7	Bohachecsky1	MS	−100	100	2
F8	Bohachecsky2	MN	−100	100	2
F9	Bohachecsky3	MN	−100	100	2
F10	Booth	MS	−10	10	2
F11	Branin	MS	−5	10	2
F12	Colville	UN	−10	10	4
F13	DixonPrice	UN	−10	10	30
F14	Easom	UN	−100	100	2
F15	Fletcher	MN	−3.1416	3.1416	2
F16	Fletcher	MN	−3.1416	3.1416	5
F17	Fletcher	MN	−3.1416	3.1416	10
F18	Griewank	MN	−600	600	30
F19	Hartman3	MN	0	1	3
F20	Hartman6	MN	0	1	6
F21	Kowalik	MN	−5	5	4
F22	Langermann	MN	0	10	2
F23	Langermann	MN	0	10	5
F24	Langermann	MN	0	10	10
F25	Matyas	UN	−10	10	2
F26	Michalewics	MS	0	3.1416	2
F27	Michalewics	MS	0	3.1416	5
F28	Michalewics	MS	0	3.1416	10
F29	Perm	MN	−4	4	4
F30	Powell	UN	−4	5	24
F31	Powersum	MN	0	4	4
F32	Quartic	US	−1.28	1.28	30
F33	Rastrigin	MS	−5.12	5.12	30
F34	Rosenbrock	UN	−30	30	30
F35	Schaffer	MN	−100	100	2
F36	Schwefel	MS	−500	500	30
F37	Schwefel_1_2	UN	−100	100	30
F38	Schwefel_2_22	UN	−10	10	30
F39	Shekel10	MN	0	10	4
F40	Shekel5	MN	0	10	4
F41	Shekel7	MN	0	10	4
F42	Shubert	MN	−10	10	2
F43	Sixhumpcamelback	MN	−5	5	2
F44	Sphere2	US	−100	100	30
F45	Step2	US	−100	100	30
F46	Stepint	US	−5.12	5.12	5
F47	Sumsquares	US	−10	10	30
F48	Trid	UN	−36	36	6
F49	Trid	UN	−100	100	10
F50	Zakharov	UN	−5	10	10

5 Results

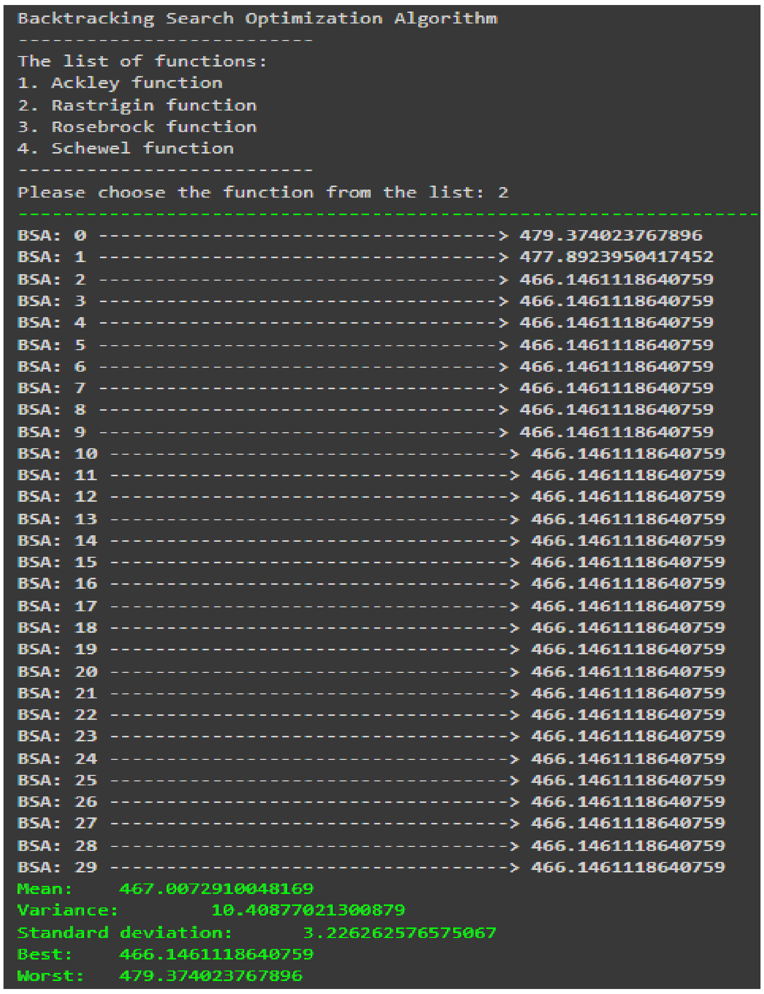
5.1 Ackley function



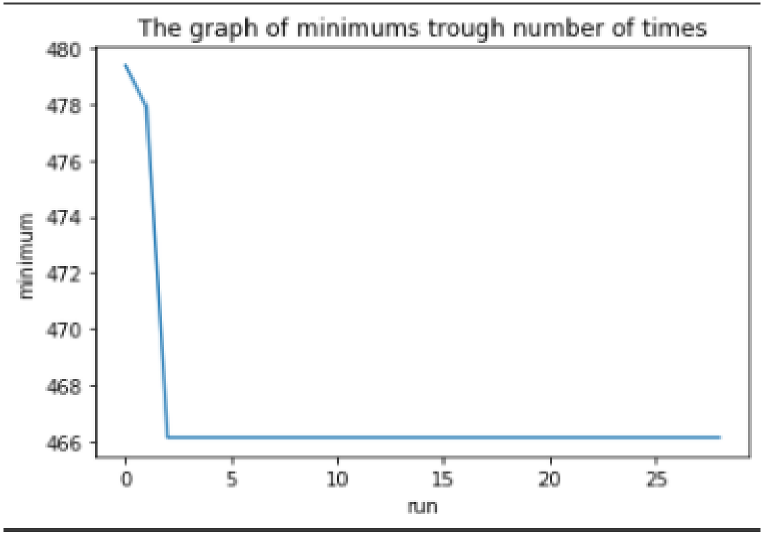
```
Backtracking Search Optimization Algorithm
--------------------------
The list of functions:
1. Ackley function
2. Rastrigin function
3. Rosebrock function
4. Schewel function
--------------------------
Please choose the function from the list: 2
------------------------------------------------------------------
BSA: 0 ----------------------------------> 434.36929290221286
BSA: 1 ----------------------------------> 410.69650441113856
BSA: 2 ----------------------------------> 410.69650441113856
BSA: 3 ----------------------------------> 410.69650441113856
BSA: 4 ----------------------------------> 410.69650441113856
BSA: 5 ----------------------------------> 410.69650441113856
BSA: 6 ----------------------------------> 410.69650441113856
BSA: 7 ----------------------------------> 410.69650441113856
BSA: 8 ----------------------------------> 410.69650441113856
BSA: 9 ----------------------------------> 410.69650441113856
BSA: 10 ----------------------------------> 410.69650441113856
BSA: 11 ----------------------------------> 410.69650441113856
BSA: 12 ----------------------------------> 410.69650441113856
BSA: 13 ----------------------------------> 410.69650441113856
BSA: 14 ----------------------------------> 410.69650441113856
BSA: 15 ----------------------------------> 410.69650441113856
BSA: 16 ----------------------------------> 410.69650441113856
BSA: 17 ----------------------------------> 410.69650441113856
BSA: 18 ----------------------------------> 410.69650441113856
BSA: 19 ----------------------------------> 410.69650441113856
BSA: 20 ----------------------------------> 410.69650441113856
BSA: 21 ----------------------------------> 410.69650441113856
BSA: 22 ----------------------------------> 410.69650441113856
BSA: 23 ----------------------------------> 410.69650441113856
BSA: 24 ----------------------------------> 410.69650441113856
BSA: 25 ----------------------------------> 410.69650441113856
BSA: 26 ----------------------------------> 410.69650441113856
BSA: 27 ----------------------------------> 410.69650441113856
BSA: 28 ----------------------------------> 410.69650441113856
BSA: 29 ----------------------------------> 410.69650441113856
Mean:     411.5128074625549
Variance:          19.324169480797927
Standard deviation:       4.395926464443864
Best:     410.69650441113856
Worst:    434.36929290221286
```



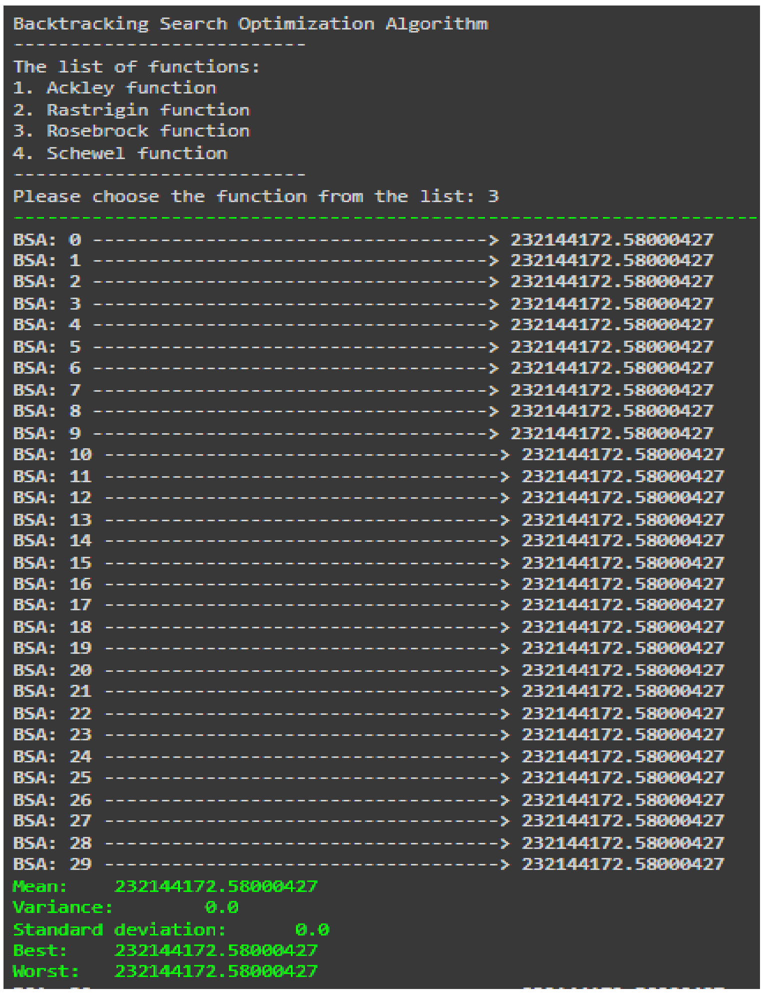
5.2 Rastrigin function



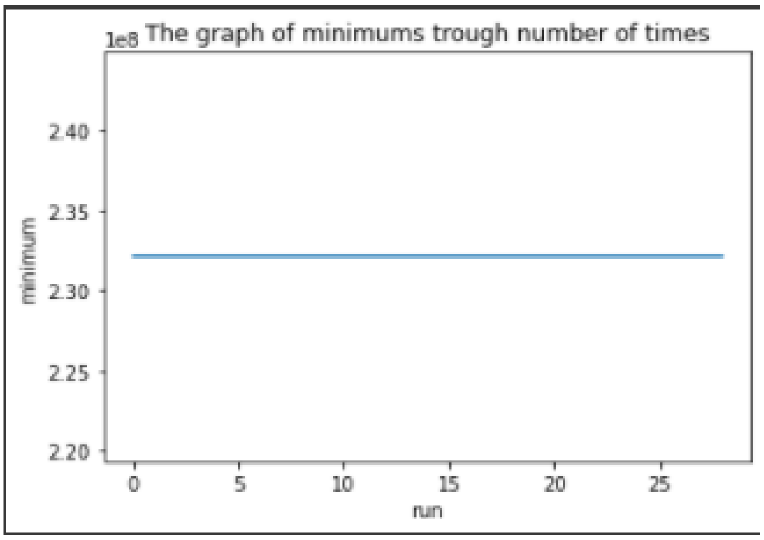
```
Backtracking Search Optimization Algorithm
--------------------------
The list of functions:
1. Ackley function
2. Rastrigin function
3. Rosebrock function
4. Schewel function
--------------------------
Please choose the function from the list: 2
-----------------------------------------------------------------
BSA: 0 -----------------------------------> 479.374023767896
BSA: 1 -----------------------------------> 477.8923950417452
BSA: 2 -----------------------------------> 466.1461118640759
BSA: 3 -----------------------------------> 466.1461118640759
BSA: 4 -----------------------------------> 466.1461118640759
BSA: 5 -----------------------------------> 466.1461118640759
BSA: 6 -----------------------------------> 466.1461118640759
BSA: 7 -----------------------------------> 466.1461118640759
BSA: 8 -----------------------------------> 466.1461118640759
BSA: 9 -----------------------------------> 466.1461118640759
BSA: 10 -----------------------------------> 466.1461118640759
BSA: 11 -----------------------------------> 466.1461118640759
BSA: 12 -----------------------------------> 466.1461118640759
BSA: 13 -----------------------------------> 466.1461118640759
BSA: 14 -----------------------------------> 466.1461118640759
BSA: 15 -----------------------------------> 466.1461118640759
BSA: 16 -----------------------------------> 466.1461118640759
BSA: 17 -----------------------------------> 466.1461118640759
BSA: 18 -----------------------------------> 466.1461118640759
BSA: 19 -----------------------------------> 466.1461118640759
BSA: 20 -----------------------------------> 466.1461118640759
BSA: 21 -----------------------------------> 466.1461118640759
BSA: 22 -----------------------------------> 466.1461118640759
BSA: 23 -----------------------------------> 466.1461118640759
BSA: 24 -----------------------------------> 466.1461118640759
BSA: 25 -----------------------------------> 466.1461118640759
BSA: 26 -----------------------------------> 466.1461118640759
BSA: 27 -----------------------------------> 466.1461118640759
BSA: 28 -----------------------------------> 466.1461118640759
BSA: 29 -----------------------------------> 466.1461118640759
Mean:     467.0072910048169
Variance:         10.40877021300879
Standard deviation:       3.226262576575067
Best:     466.1461118640759
Worst:    479.374023767896
```



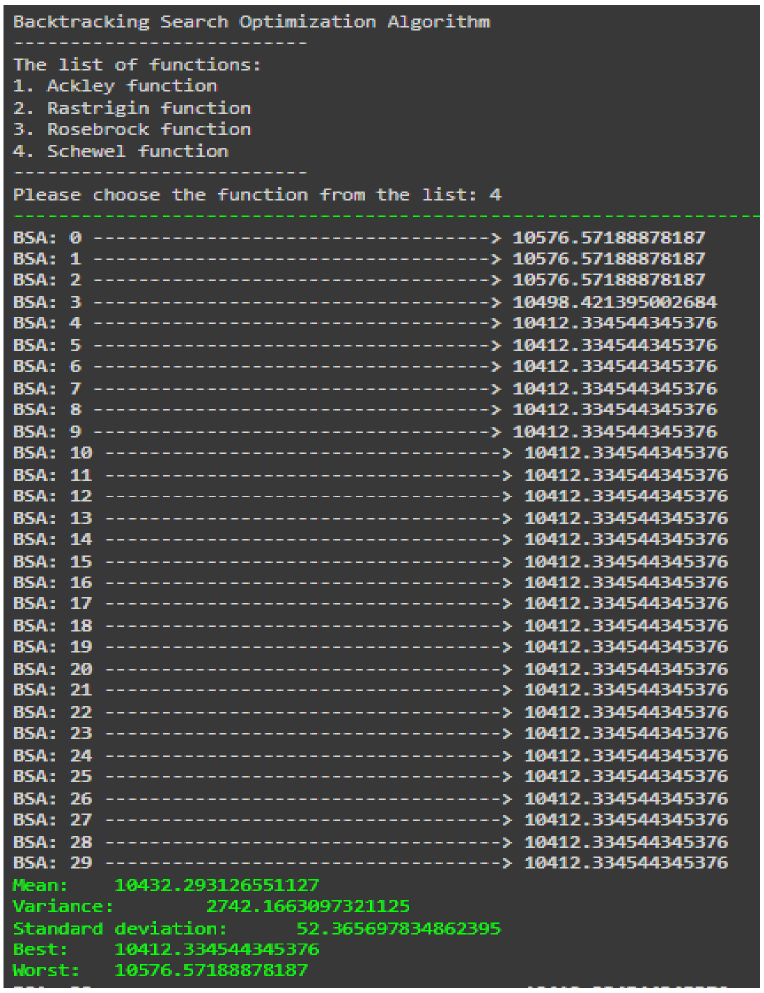
5.3 Rosenbrock function



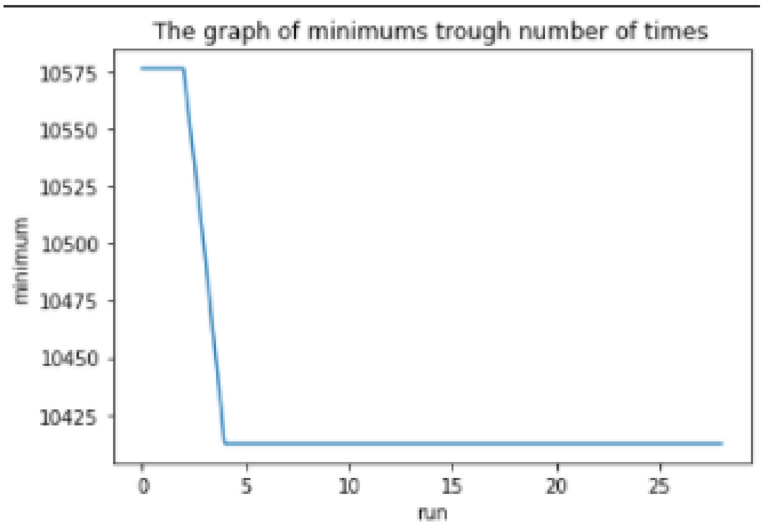
```
Backtracking Search Optimization Algorithm
--------------------------
The list of functions:
1. Ackley function
2. Rastrigin function
3. Rosebrock function
4. Schewel function
--------------------------
Please choose the function from the list: 3
------------------------------------------------------------------
BSA: 0 -----------------------------------> 232144172.58000427
BSA: 1 -----------------------------------> 232144172.58000427
BSA: 2 -----------------------------------> 232144172.58000427
BSA: 3 -----------------------------------> 232144172.58000427
BSA: 4 -----------------------------------> 232144172.58000427
BSA: 5 -----------------------------------> 232144172.58000427
BSA: 6 -----------------------------------> 232144172.58000427
BSA: 7 -----------------------------------> 232144172.58000427
BSA: 8 -----------------------------------> 232144172.58000427
BSA: 9 -----------------------------------> 232144172.58000427
BSA: 10 -----------------------------------> 232144172.58000427
BSA: 11 -----------------------------------> 232144172.58000427
BSA: 12 -----------------------------------> 232144172.58000427
BSA: 13 -----------------------------------> 232144172.58000427
BSA: 14 -----------------------------------> 232144172.58000427
BSA: 15 -----------------------------------> 232144172.58000427
BSA: 16 -----------------------------------> 232144172.58000427
BSA: 17 -----------------------------------> 232144172.58000427
BSA: 18 -----------------------------------> 232144172.58000427
BSA: 19 -----------------------------------> 232144172.58000427
BSA: 20 -----------------------------------> 232144172.58000427
BSA: 21 -----------------------------------> 232144172.58000427
BSA: 22 -----------------------------------> 232144172.58000427
BSA: 23 -----------------------------------> 232144172.58000427
BSA: 24 -----------------------------------> 232144172.58000427
BSA: 25 -----------------------------------> 232144172.58000427
BSA: 26 -----------------------------------> 232144172.58000427
BSA: 27 -----------------------------------> 232144172.58000427
BSA: 28 -----------------------------------> 232144172.58000427
BSA: 29 -----------------------------------> 232144172.58000427
Mean:     232144172.58000427
Variance:         0.0
Standard deviation:       0.0
Best:     232144172.58000427
Worst:    232144172.58000427
```



5.4 Schwefel function



```
Backtracking Search Optimization Algorithm
--------------------------
The list of functions:
1. Ackley function
2. Rastrigin function
3. Rosebrock function
4. Schewel function
--------------------------
Please choose the function from the list: 4
-------------------------------------------------------------------
BSA: 0 -----------------------------------> 10576.57188878187
BSA: 1 -----------------------------------> 10576.57188878187
BSA: 2 -----------------------------------> 10576.57188878187
BSA: 3 -----------------------------------> 10498.421395002684
BSA: 4 -----------------------------------> 10412.334544345376
BSA: 5 -----------------------------------> 10412.334544345376
BSA: 6 -----------------------------------> 10412.334544345376
BSA: 7 -----------------------------------> 10412.334544345376
BSA: 8 -----------------------------------> 10412.334544345376
BSA: 9 -----------------------------------> 10412.334544345376
BSA: 10 -----------------------------------> 10412.334544345376
BSA: 11 -----------------------------------> 10412.334544345376
BSA: 12 -----------------------------------> 10412.334544345376
BSA: 13 -----------------------------------> 10412.334544345376
BSA: 14 -----------------------------------> 10412.334544345376
BSA: 15 -----------------------------------> 10412.334544345376
BSA: 16 -----------------------------------> 10412.334544345376
BSA: 17 -----------------------------------> 10412.334544345376
BSA: 18 -----------------------------------> 10412.334544345376
BSA: 19 -----------------------------------> 10412.334544345376
BSA: 20 -----------------------------------> 10412.334544345376
BSA: 21 -----------------------------------> 10412.334544345376
BSA: 22 -----------------------------------> 10412.334544345376
BSA: 23 -----------------------------------> 10412.334544345376
BSA: 24 -----------------------------------> 10412.334544345376
BSA: 25 -----------------------------------> 10412.334544345376
BSA: 26 -----------------------------------> 10412.334544345376
BSA: 27 -----------------------------------> 10412.334544345376
BSA: 28 -----------------------------------> 10412.334544345376
BSA: 29 -----------------------------------> 10412.334544345376
Mean:    10432.293126551127
Variance:        2742.1663097321125
Standard deviation:      52.365697834862395
Best:    10412.334544345376
Worst:   10576.57188878187
```



6 Conclusion

In conclusion, BSA is a new EA with metaheuristic approach for solving optimization problems. There are 5 stages to implement BSA:

1. Initialization
2. Selection-I
3. Mutation
4. Crossover
5. Selection-II

UML diagrams, such as Use-Case diagram and Class diagram are created to make implementation part much easier.

Python programming language was used in order to implement all the stages mentioned above and the whole algorithm. Specifically NumPy library was very handy in terms of calculations.

Results are obtained by using different performance test functions such Ackley, Rosenbrock, Ratrigin and Schwefel.

Bibliography

[1] P. Civicioglu, ”Backtracking search optimization algorithm for numerical optimization problems,” Applied Mathematics and Computation, vol. 219, no. 15, 2013.

[2] Dr. Ahmed Fouad Ali ”Backtracking Search Optimization Algorithm (BSA)”

[3] Yashpal Sheoran, ”Development of Backtracking Search Optimization Algorithm Toolkit in LabVIEW™”, ”Procedia Computer Science”, vol. 57, 2015