Application of Differential Evolution Algorithm in Solving Mathematical Benchmark Functions

Ravan Iskandarov

Haji Akhundzada

Arzuman Hasanov

Abstract:

This project explores the application of metaheuristic algorithms to optimize mathematical benchmark functions. Specifically, the project investigates the performance of Differential Evolution (DE), considering benchmark functions like the Sphere, Ackley, Rastrigin, Rosenbrock, and Schwefel functions. The DE algorithm is implemented to minimize these functions, analyzing its convergence properties and effectiveness in finding global minima. The study aims to compare and evaluate the DE algorithm's performance across diverse benchmark functions, shedding light on its adaptability and efficiency in solving complex optimization problems.

Keywords:
Metaheuristics, Benchmark Functions, Optimization, Differential Evolution Algorithm

Introduction:

Optimization is a fundamental concept permeating various fields, serving as a cornerstone for enhancing efficiency, resource utilization, and problem-solving across diverse domains. It involves the quest for finding the best possible solution among a multitude of alternatives, a pursuit central to decision-making processes in engineering, science, economics, and beyond. The significance of optimization lies in its ability to streamline operations, minimize costs, maximize performance, and address intricate real-world problems.

However, in the realm of complex and nonlinear problems, traditional optimization approaches often encounter limitations, struggling with high-dimensional spaces, non-convex landscapes, and multimodal functions. This is where metaheuristic algorithms emerge as powerful tools, providing innovative solutions to circumvent the limitations of classical optimization methods.

Metaheuristic algorithms represent a class of optimization techniques inspired by natural phenomena, social behaviors, or mathematical principles. Their flexibility, adaptability, and ability to explore vast solution spaces make them indispensable in addressing challenging optimization tasks. These algorithms, including Differential Evolution (DE), Genetic Algorithms (GA), Particle Swarm Optimization (PSO), among others, offer heuristic strategies capable of navigating complex landscapes, enabling the discovery of optimal or near-optimal solutions.

Benchmark functions serve as standardized tools for evaluating and comparing the performance of optimization algorithms. These functions simulate diverse problem landscapes, each posing unique challenges to optimization algorithms. Functions like Sphere, Ackley, Rastrigin, Rosenbrock, and Schwefel are commonly used benchmarks due to their varying complexities, multimodalities, and non-linearity, making them ideal testbeds to assess the efficacy and robustness of optimization algorithms.

In this study, the focus lies on the DE algorithm, exploring its performance across these benchmark functions. Through comprehensive analysis and experimentation, this research aims to shed light on the adaptability, convergence behavior, and efficiency of the DE algorithm in solving intricate optimization problems posed by these diverse benchmark functions.

Literature Review:

Differential Evolution (DE) has garnered substantial attention in the field of optimization owing to its effectiveness in solving a wide array of optimization problems. Originating from the evolutionary computation domain, DE is a population-based metaheuristic algorithm that mimics the process of natural evolution to iteratively search and converge towards optimal solutions.

DE operates on a population of candidate solutions, often referred to as individuals or vectors, evolving them through mutation, crossover, and selection. The algorithm's mutation strategy, typically based on the difference between randomly selected individuals, aids in exploration, while crossover and selection mechanisms facilitate exploitation to refine solutions.

Numerous studies have showcased the efficacy of DE in solving optimization problems across various domains. Research by Storn and Price (1997) introduced DE as a powerful and efficient optimizer, demonstrating its superiority in solving complex numerical optimization problems compared to other evolutionary algorithms.

DE's success lies in its ability to handle problems with high dimensionality, nonlinearity, and multimodality. Empirical studies by Das and Suganthan (2011) compared DE with other evolutionary algorithms, highlighting its competitive performance in optimization tasks, particularly in high-dimensional spaces.

Regarding benchmark functions, the selection of suitable test problems plays a crucial role in evaluating and comparing optimization algorithms. Functions like Sphere, Ackley, Rastrigin, Rosenbrock, and Schwefel are commonly employed as standard benchmarks due to their diverse properties—ranging from smooth and convex (Sphere) to rugged and multimodal (Rastrigin). Each function challenges optimization algorithms in unique ways, assessing their ability to navigate various landscapes and locate global optima efficiently.

Research on benchmark functions aims to comprehensively analyze and compare the performance of optimization algorithms. Numerous studies by Eiben et al. (1999), Liang et al. (2013), and others have employed these benchmark functions to evaluate the convergence rate, accuracy, and robustness of optimization algorithms, providing insights into their strengths and weaknesses across different problem landscapes.

This literature review underscores the significance of DE as a versatile optimization tool and highlights the relevance of benchmark functions in assessing algorithm performance, laying the groundwork for evaluating DE's efficacy across diverse mathematical landscapes in this study.

Problem Definition:

The report focuses on four widely acknowledged mathematical benchmark functions: Ackley Function, Rosenbrock Function, Schwefel Function, and Rastrigin Function. These functions serve as standardized tests to evaluate the performance of optimization algorithms due to their diverse landscapes and varying complexities.

1. Ackley Function:

The Ackley function, a widely recognized benchmark, is described by the equation:

$$f(x)=-a\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right)-\exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right)+a+\exp(1)$$

Recommended variable values are: a = 20, b = 0.2 and c = 2π

Where:

- $x=(x_1,x_2,\ldots,x_d)$ represents the vector of d variables within the search space.
- The function operates in d dimensions.
- The global minimum of $f(x)=0$ occurs at $x=(0,0,\ldots,0)$.

Characteristics:

Non-Convexity: Ackley's function presents multiple local minima, challenging optimization algorithms to differentiate between the global and local optima.

Complex Landscape: The landscape exhibits oscillatory behavior and intricate topology, complicating the search for the global minimum.

Hypercube Domain: Often evaluated within the hypercube.

$x_i \in [-32.768, 32.768]$ for all $i=1,2,\ldots,d$, though it can vary based on requirements.

2. Rastrigin Function:

The Rastrigin function is defined as:

$$f(x) = Ad + \sum_{i=1}^{d} [x_i^2 - A\cos(2\pi x_i)]$$

Where:

- $x=(x_1,x_2,\ldots,x_d)$ represents the d-dimensional input vector.
- A is a constant term (commonly $A=10$).
- The function operates in d dimensions.

Key Characteristics:

Non-Convexity: Rastrigin's landscape is highly non-convex, containing many local minima, which poses challenges for optimization algorithms.

Periodic Structure: The function exhibits periodicity due to the cosine term, creating a rugged landscape with regularly spaced local minima.

Global Minimum: The global minimum of $f(x)=0$ is attained at $x=(0,0,\ldots,0)$.

Domain:

The function is typically evaluated within a bounded domain, commonly in the range $x_i \in [-5.12, 5.12]$ for all $i=1,2,\ldots,d$ though this range can vary based on requirements.

3. Rosenbrock Function:

The Rosenbrock function, also known as the Rosenbrock's valley or banana function, is described as:

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

Where:

- $x=(x_1,x_2,\ldots,x_d)$ represents the d-dimensional input vector.
- The function operates in d dimensions.

Key Characteristics:

Non-Convexity: The Rosenbrock function is highly non-convex, featuring a curved, elongated valley with a single narrow global minimum.

Narrow Valley: Its valley-like structure poses challenges for optimization algorithms, as they struggle to navigate the narrow, elongated path to the global minimum.

Global Minimum: The global minimum of $f(x)=0$ is located at $x=(1,1,\ldots,1)$.

Domain:

It is typically evaluated within a bounded domain, often in the range $x_i\in[-5,5]$ for all $i=1,2,\ldots,d$ although this range can be adjusted based on specific requirements.

4. Schwefel Function:

The Schwefel function is represented as:

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

Where:

- $x=(x_1,x_2,\ldots,x_d)$ represents the d-dimensional input vector.
- The function operates in d dimensions.

Key Characteristics:

High Non-Linearity: The Schwefel function is highly non-linear with a rugged landscape, featuring many local minima and complex structures.

Global Minimum: The global minimum of the function is located at f(x)=0 when x=(0,0,…,0).

Sensitivity to Local Optima: It exhibits sensitivity to initial conditions due to the presence of numerous local minima.

Domain:

The function is typically evaluated within a bounded domain, commonly in the range $x_i \in [-500,500]$ for all i=1,2,…,d, though this range can vary based on requirements.

Solution Methods:

Mathematical Understanding of Crossover Effects:

Let's present basic terminology to comprehend the impacts of binomial and exponential crossover over numerous generations. Let ${U_i}^{GN=k}$ represent a vector in the set of trial vectors, where k parameters are inherited from the mutant vector ${V_i}^{(G)}$. Additionally, $P({U_i}^{GN=k})$ denotes the probability of generating such a trial vector candidate with k parameters inherited from mutant vector ${V_i}^{(G)}$ .The probability $P({U_i}^{GN=k})$ can be calculated using exponential crossover and binomial crossover, denoted as *Cr*, for a *D*-dimensional optimization problem:

$$P(U_i^{(GN=k)}) = \begin{cases} Cr^{k-1} \times (1 - Cr), & \text{if } k < D \\ Cr^{D-1}, & \text{if } k = D \end{cases}$$

$$(D-1)! = \prod_{i=1}^{D-1} i \quad \text{and} \quad C_D^{D-1} = \frac{(D-1)!}{(k-1)!(D-k)!}$$

For both exponential and binomial crossover, the expectation that N parameters in the trial vector are inherited from the mutant vector may now be computed:

$$E(N) = \frac{1 - Cr^D}{1 - Cr}$$

$$E(N) = (D - 1) \times Cr + 1$$

For a given binomial crossover rate C_{r_b}, there exists an exponential crossover rate C_{r_e}, ∈ [0,1] satisfying the equation:

$$\frac{1 - Cr_e^D}{1 - Cr_e} = (D - 1) \times Cr_b + 1$$

From this angle, by figuring out the appropriate crossover rate Cr and its associated parameters, a DE version using exponential crossover can compete in numerical optimization. On the other hand, exponential crossover DE algorithms typically outperform binomial crossover algorithms in most optimization applications, and most DE researchers currently feel that DE variants with exponential crossover are better suited for optimization problems with linkages among neighboring variables. Although the properties of exponential crossover have been discussed, not generally recognized and competitive DE variant with exponential crossover for numerical optimization has been found.

Differential Evolution (DE) Mutation Operator

The mutation operator is used in the Differential Evolution (DE) method to produce a trial vector for every member of the current population. In this procedure, a weighted differential is used to mutate a target vector. The trial vector, in turn, is utilized by the crossover operator to produce offspring.

For each parent, denoted as $x_i(t)$, the trial vector $u_i(t)$ is generated through the following steps:

1. **Select a Target Vector:**
   - Randomly choose an index i_1 from the population, ensuring that $i \neq i_1$.
2. **Randomly Choose Two Distinct Individuals:**
   - Select two distinct indices i_2 and i_3 from the population, such that $i \neq i_1 \neq i_2 \neq i_3$.
   - The indices i_2 and i_3 are uniformly sampled from the range $U(1, ns)$, where ns is the population size.

The trial vector is computed by perturbing the target vector according to the formula:

$$u_i(t) = x_{i1}(t) + \beta \times (x_{i2}(t) - x_{i3}(t))$$

Here, β, belonging to the interval $(0, \infty)$, serves as the scale factor governing the amplification of the differential variation.

Pseudocode for Differential Evolution:

Detailed steps of the Differential Evolution Algorithm are provided along with relevant pseudocode.

1. Generate a population of potential solutions randomly within the defined search space

2. Compute the fitness values for each individual in the population using the defined objective function.

3. While the termination condition remains unsatisfied, do the following steps:

4. Create mutant vectors using a mutation strategy in Eq. 1

5. Create trial vectors by recombining noisy vectors with parents vectors according to Eq. 2

6. Assess the trial vectors' fitness by computing their corresponding objective function values.

7. Select wining vectors according to Eq 3. as individuals in the new generation

8. End While

$$V_{i,g} = X_{r_1,g} + F \times (X_{r_2,g} - X_{r_3,g}) \qquad (1)$$

where $V_{i,g}$ represents the mutant vector, i stands for the index of the vector, g stands for the generation, $r_1, r_2, r_3 \in \{ 1,2,\ldots,N_p\}$ are random integers and F is the scaling factor in the interval [0, 2].

(eq. 1)

$$T_{i,g}[j] = \begin{cases} V_{i,g}[j] & \text{if } rand[0,1] < CR \quad \text{or} \quad j = j_{rand} \\ X_{i,g}[j] & \text{otherwise} \end{cases} \qquad (2)$$

where j stands for the index of every parameter in a vector, J_{rand} is a randomly selected integer between 1 and N_p to ensure that at least one parameter from mutant vector will enter the trial vector and CR is the probability of recombination.

(eq. 2)

$$X_{i,g+1} = \begin{cases} T_{i,g} & \text{if } f(T_{i,g}) < f(X_{i,g}) \\ X_{i,g} & \text{otherwise} \end{cases} \qquad (3)$$

where $T_{i,g}$ is the trial vector, $X_{i,g}$ is an individual in the population, $X_{i,g+1}$ is the individual in the next generation, $f(T_{i,g})$ represents the fitness value of the trial vector and $f(X_{i,g})$ stands for the fitness value of the individual in the population.

(eq. 3)

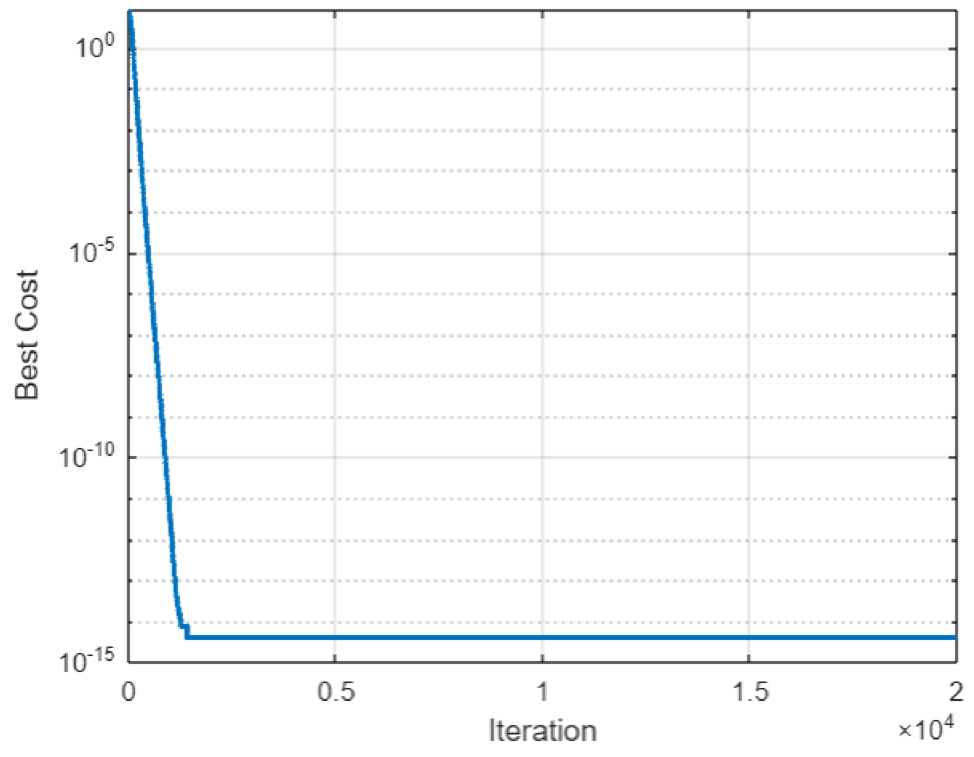
Results:

The Differential Evolution Algorithm was applied to optimize the benchmark functions across thirty runs. Objective values such as best, worst, average, and standard deviation were recorded, alongside computational time. Results obtained from MATLAB-implemented algorithms for these benchmarks were also gathered for comparison.

Here is obtained results for Differential Evolution Algorithm:

1. Ackley Function**:**

MATLAB results:
Best Cost: 3.9968e-15
Worst Cost: 7.5495e-15
Mean Cost: 5.6547e-15
Standard Deviation: 1.8027e-15



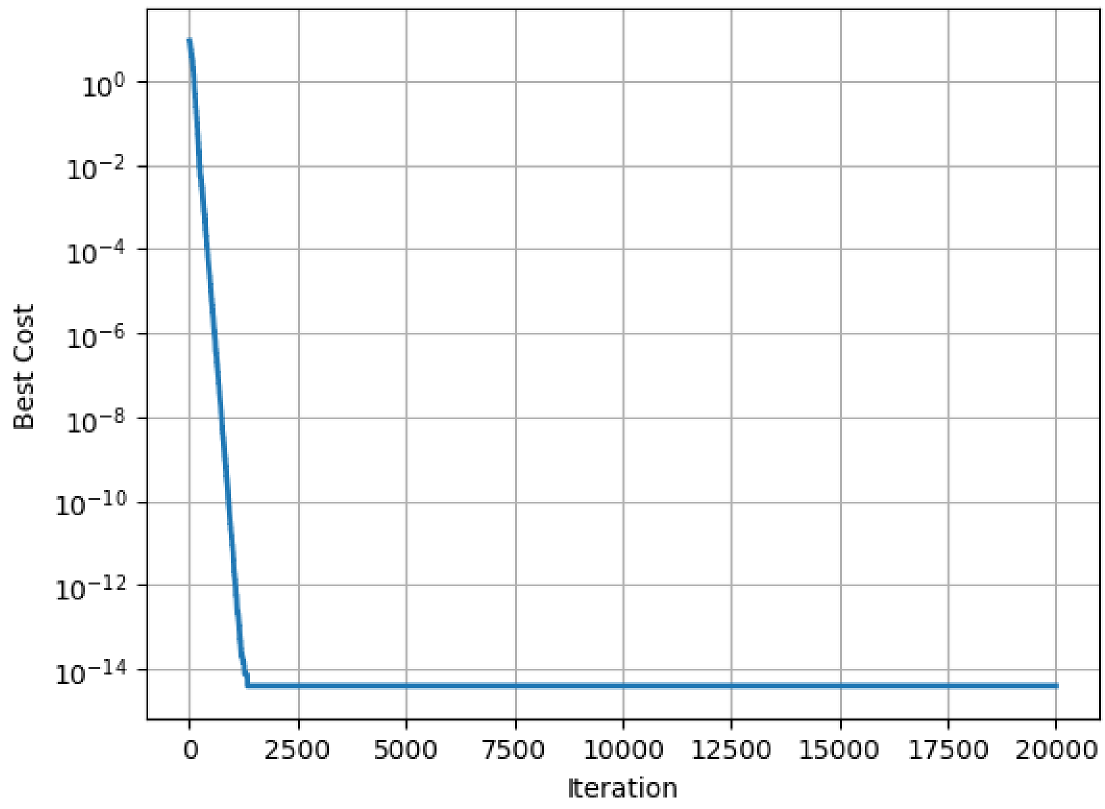
Python results:
Best Minimum: 3.9968028886505635e-15

Worst Minimum: 9.1741783632068

Average Minimum: 0.023796583079477927

Standard Deviation of Minimums: 0.3494273622131332

Computational Time: 48.855628490448 seconds



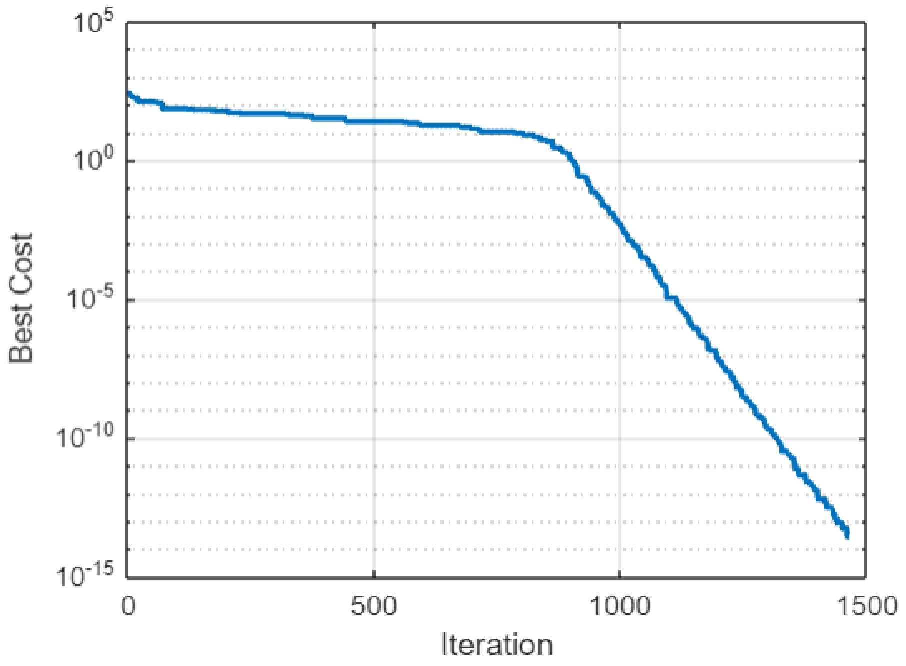
2. Rastrigin Function:

MATLAB results:
Mean of Global Minimum Values: 0.033165301903109894

Standard Deviation of Global Minimum Values: 0.1786006166266184

Best (Minimum) Global Minimum Value: 0.0

Worst (Maximum) Global Minimum Value: 0.9949590570932969



Python results:

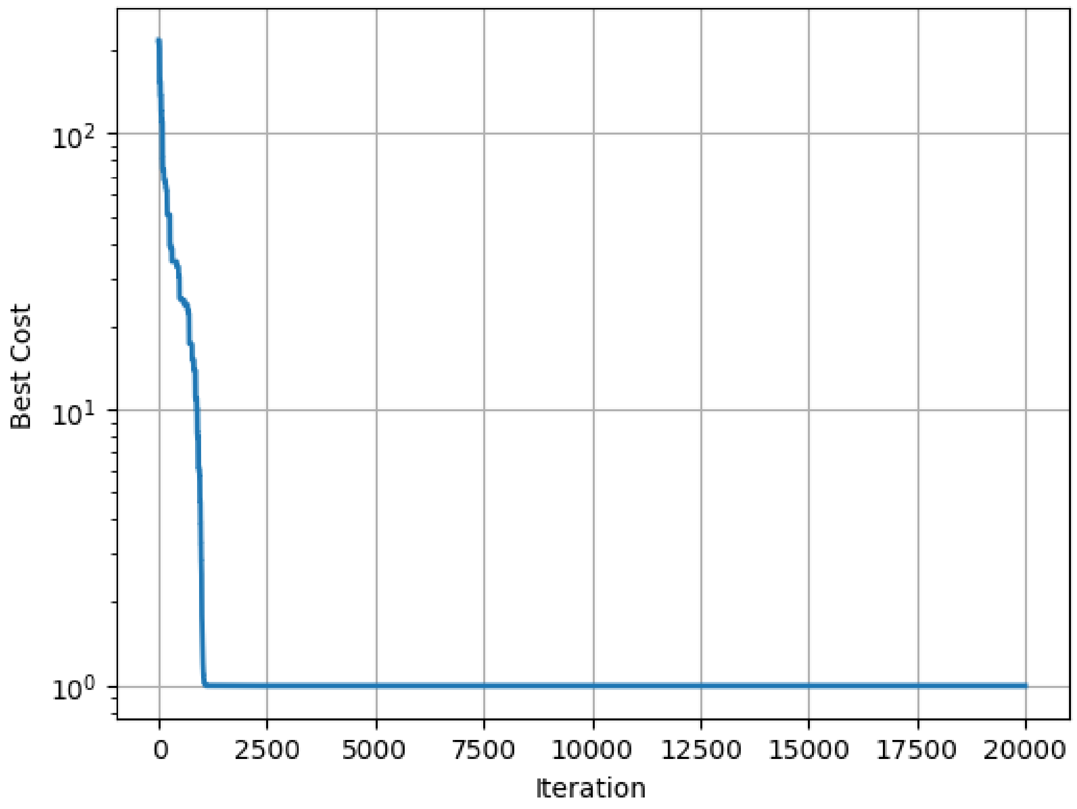
Best Minimum: 0.9949590570932969

Worst Minimum: 217.94608778187057

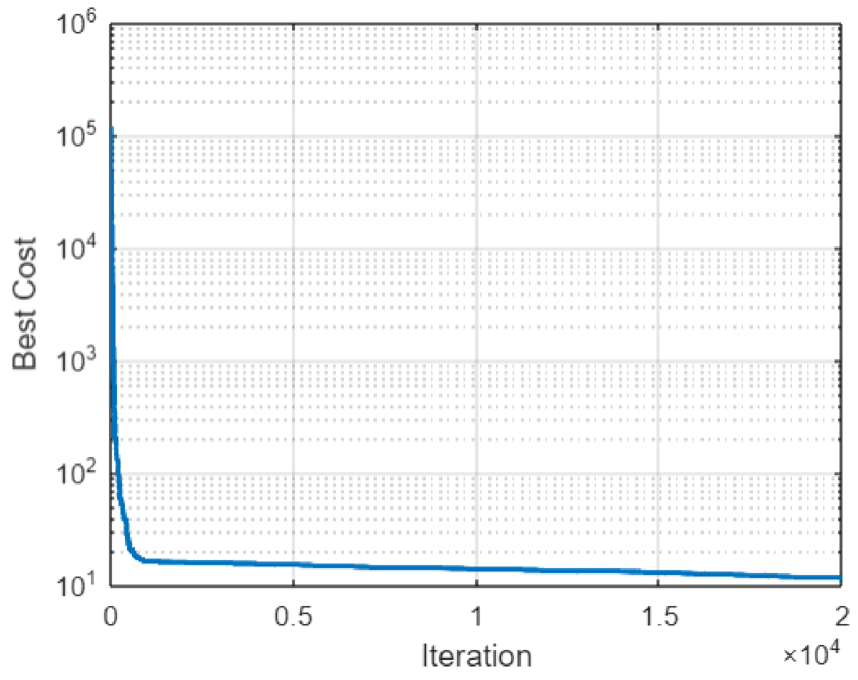
Average Minimum: 2.887562516695333

Standard Deviation of Minimums: 11.795843201442722

Computational Time: 42.51223587989807 seconds



3. Rosenbrock Function**:**
MATLAB results:
Best Cost: 8.2912
Worst Cost: 67.1501
Mean Cost: 22.6326
Standard Deviation: 23.5769



Python results:

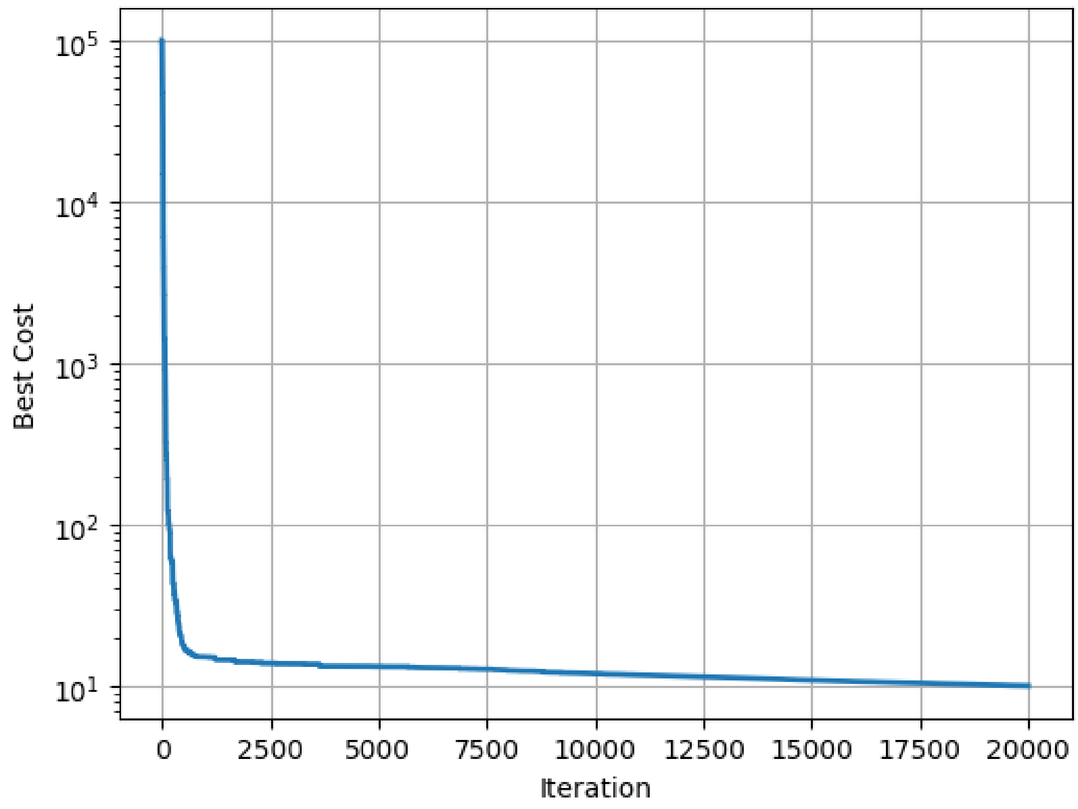
Best Minimum: 10.042253899031554

Worst Minimum: 100020.99443910777

Average Minimum: 82.20887469483301

Standard Deviation of Minimums: 2038.637564243378

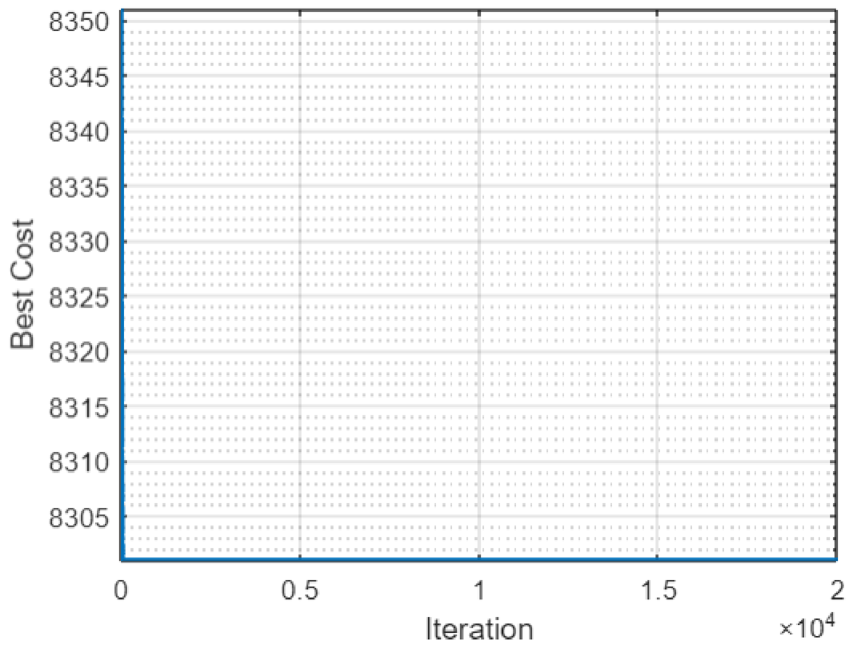
Computational Time: 50.69840121269226 seconds



4. Schwefel Function**:**

MATLAB results:

Best Cost: 8300.9831
Worst Cost: 8300.9831
Mean Cost: 8300.9831
Standard Deviation: 5.5503e-12



Python results:

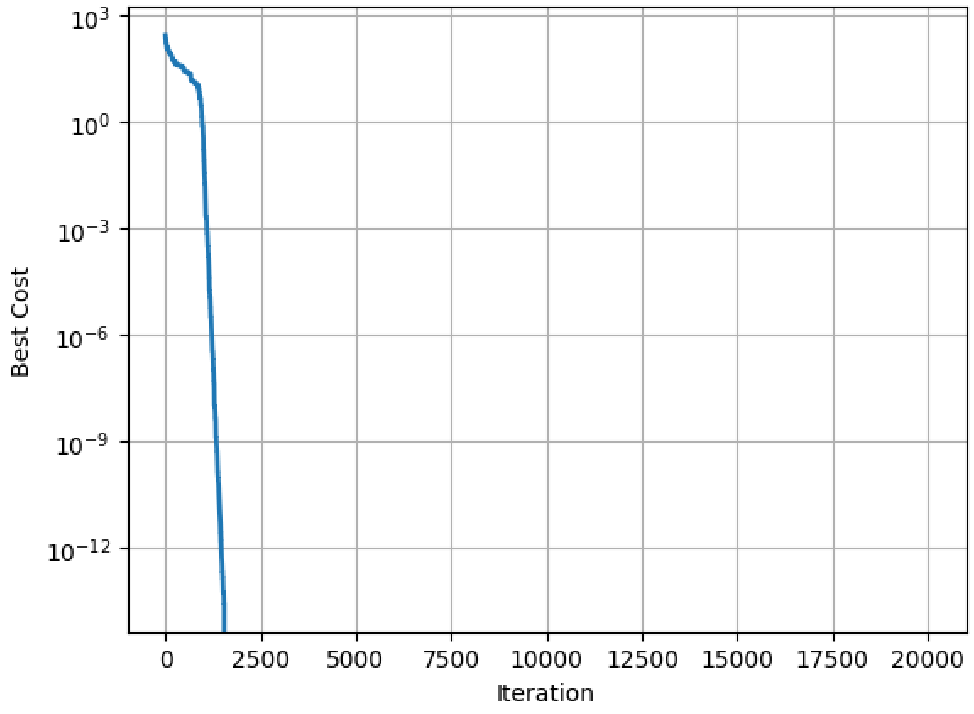
Best Minimum: 0.0

Worst Minimum: 267.17806637455203

Average Minimum: 1.9883150028334577

Standard Deviation of Minimums: 12.550198166164343

Computational Time: 51.82969903945923 seconds

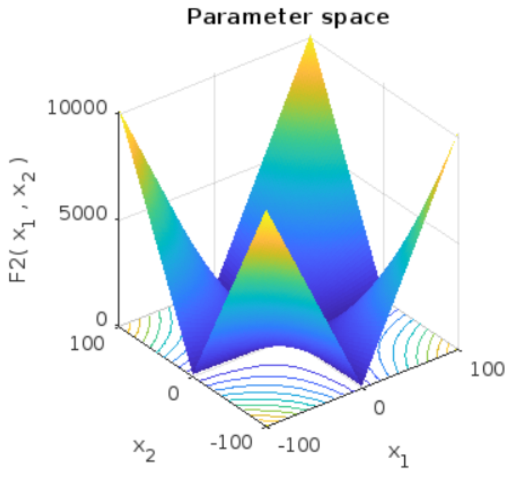


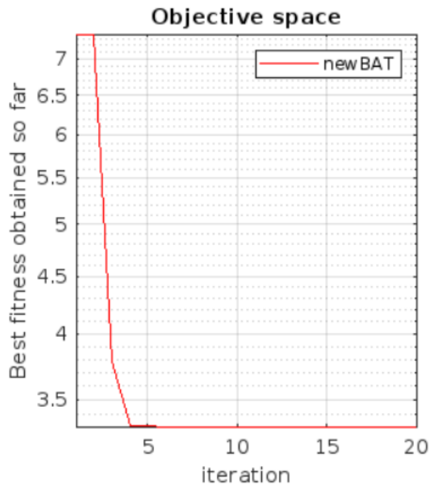
Matlab results of other algorithms:

1. Bat Optimization Algorithm(BOA):

Ackley Function:

The best solution obtained by BAT is : [0.20906 -0.16433 0.43977 0.18191 -0.048602 -0.48054 0.56312 -0.37298 0.84803 -0.0051947]



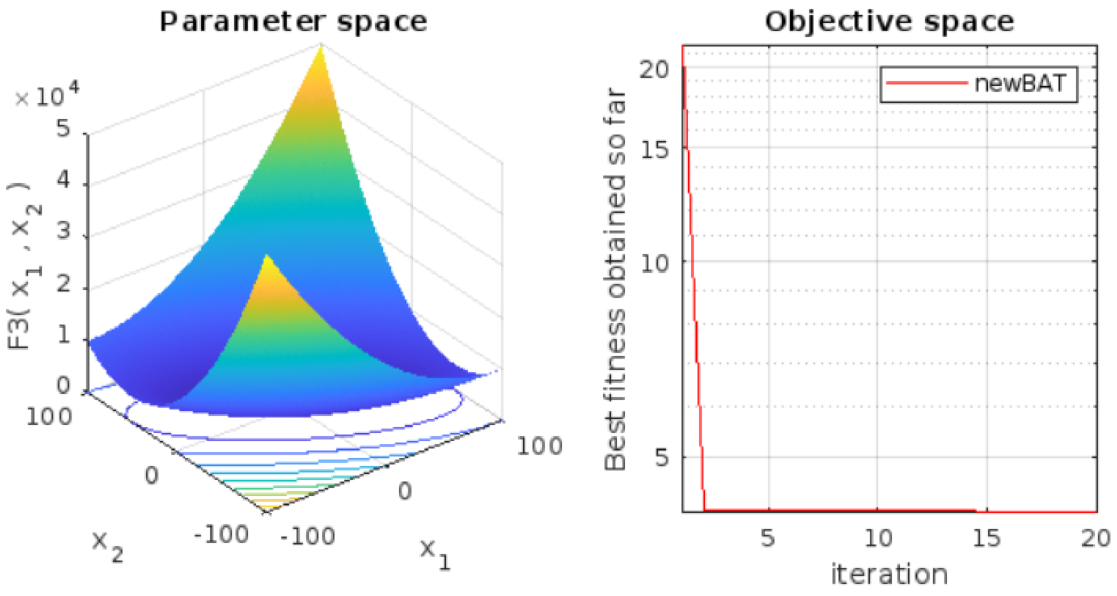


The best optimal value of the objective funciton found by BAT is : 3.3135

Rastrigin Function:

The best solution obtained by BAT is : [-0.56118   -0.55973   -0.12861   0.90442   1.0177   -0.16452   -0.49504   -0.039376   -0.092128   0.49424]

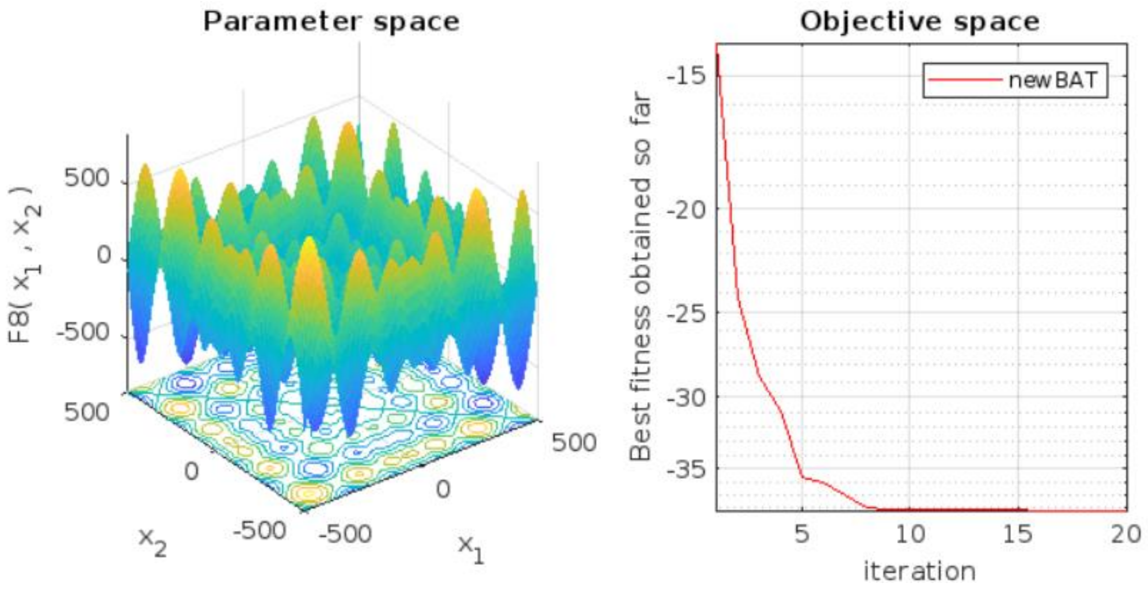
The best optimal value of the objective funciton found by BAT is : 4.1184



Rosenbrock Function:

The best solution obtained by BAT is : [4.0307   5.2719   6.0307   4.7051   6.0307   6.0307   4.0307   5.7746   5.7273   4.7394]

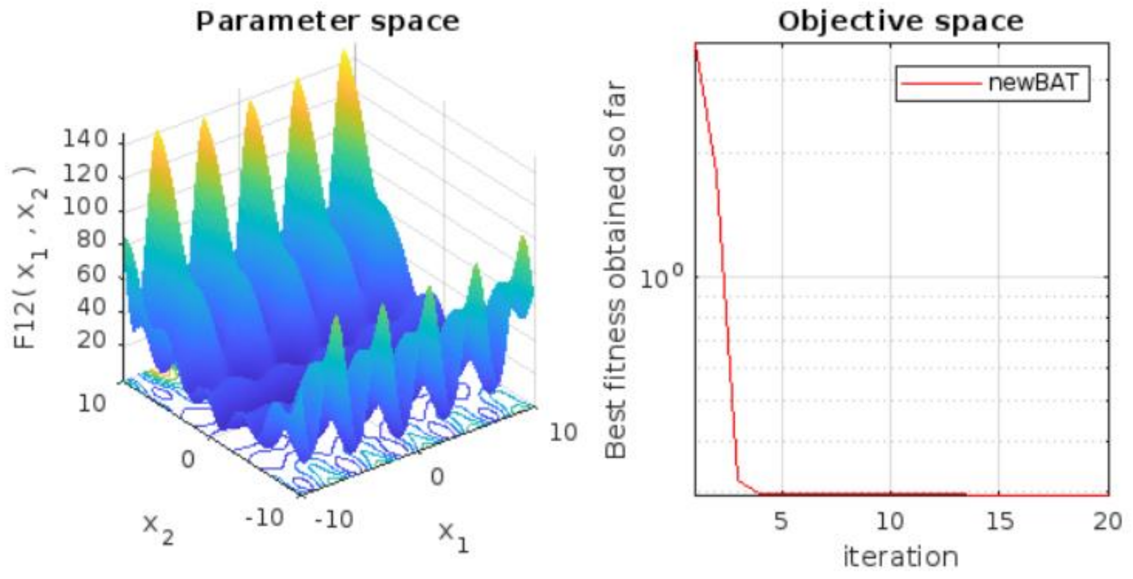
The best optimal value of the objective funciton found by BAT is : -38.2751



Schwefel Function:

The best solution obtained by BAT is : [-1.1224   -0.79318   -0.83297   -0.052887   -0.84168   -1.0322   0.46961   -1.2743   -0.21355   0.60752]

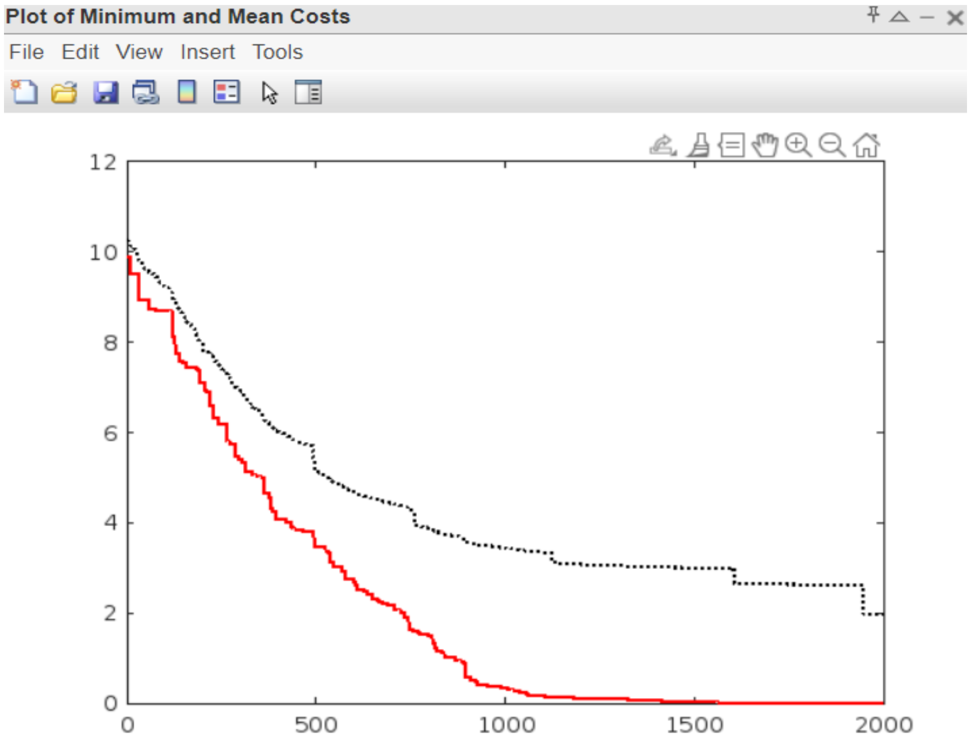
The best optimal value of the objective funciton found by BAT is : 0.29576



2. Imperialist Competitive Algorithm(ICA):

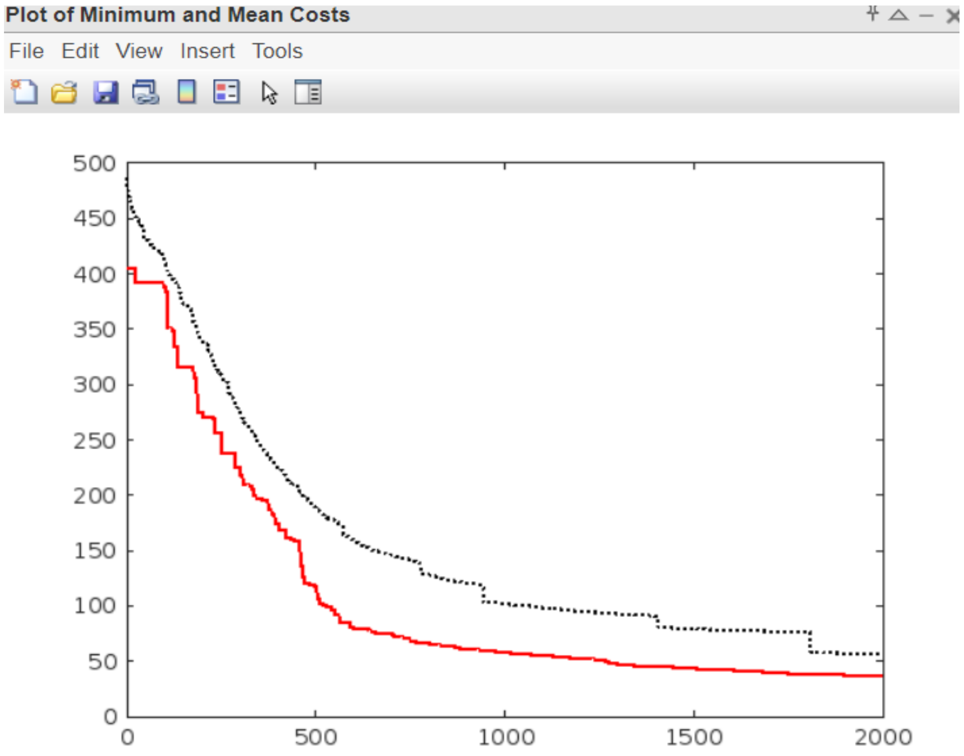
Ackley Function:

ans = 0.0063



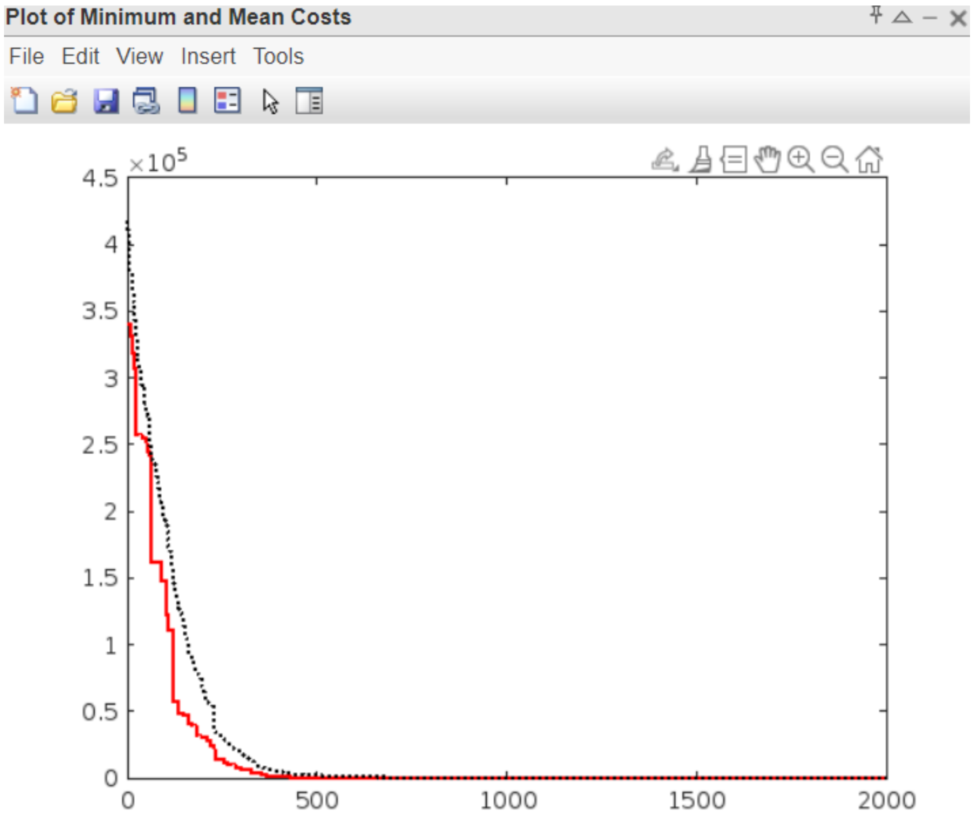
Rastrigin Function:

ans = 37.2961



Rosenbrock Function:

ans = 6.4090



3. Harmony Search Algorithm(HSA):

Ackley Function:

Best Solution: [1.61796349580918e-05 5.92158412758217e-05 4.95137710002687e-06 -1.08576311551071e-05]

Best Fitness Value: 0.0001

Rastrigin Function:

Best Solution: [5.08139354275737e-05 6.05171788260926e-06 -3.39230545617391e-05 1.08823314693944e-06]

Best Fitness Value: 0.0000

Rosenbrock Function:

Best Solution: [0.980812616474601 0.962007644617358 0.925333125171982 0.855649003338888]

Best Fitness Value: 0.0074

Schwefel Function:

Best Solution: [420.916893756351 422.120966561246 421.012165100039 420.95341456009]

Best Fitness Value: -1675.7634

4. Firefly Optimization algorithm (FOA):

Ackley Function:

Best Fitness: 0.000000

Worst Fitness: 3.508529

Average Fitness: 0.202515

Standard Deviation Fitness: 0.602322

Rastrigin Function:

Best Fitness: 0.000000

Worst Fitness: 7.652042

Average Fitness: 0.488921

Standard Deviation Fitness: 1.199635

Rosenbrock Function:

Best Fitness: 0.000000

Worst Fitness: 16.880556

Average Fitness: 0.167216

Standard Deviation Fitness: 0.989674

Schwefel Function:

Best Fitness: -2360.908931

Worst Fitness: 285.179307

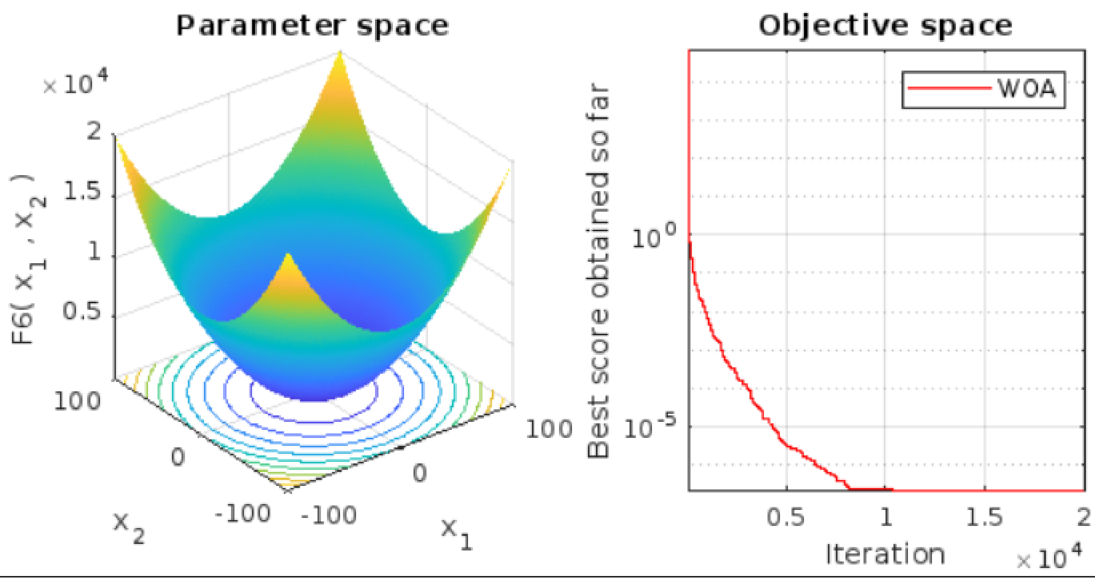
Average Fitness: -32.824971

Standard Deviation Fitness: 290.195134

5. Whale Optimization Algorithm (WOA):

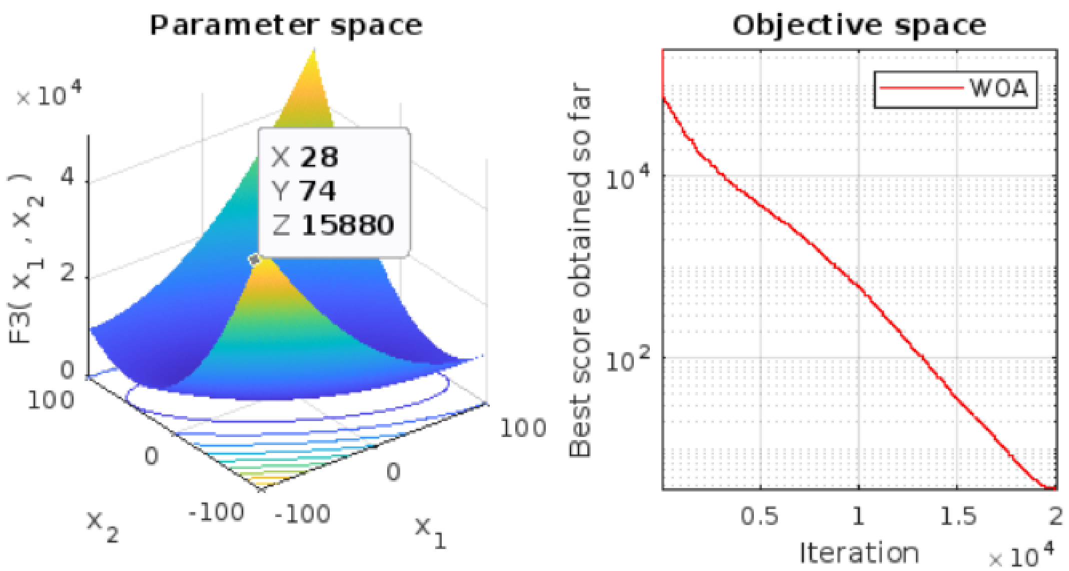
Ackley Function:

The best solution obtained by WOA is : [-0.50013 -0.49989 -0.49995 -0.49998 -0.49999 -0.49986 -0.50012 -0.50002 -0.49996 -0.50004 -0.50004 -0.49983 -0.50002 -0.50013 -0.49995 -0.49994 -0.49987 -0.49997 -0.50006 -0.49993 -0.50011 -0.50003 -0.5001 -0.50001 -0.50007 -0.49998 -0.5001 -0.5 -0.50008 -0.49984]
The best optimal value of the objective function found by WOA is : 2.1999e-07



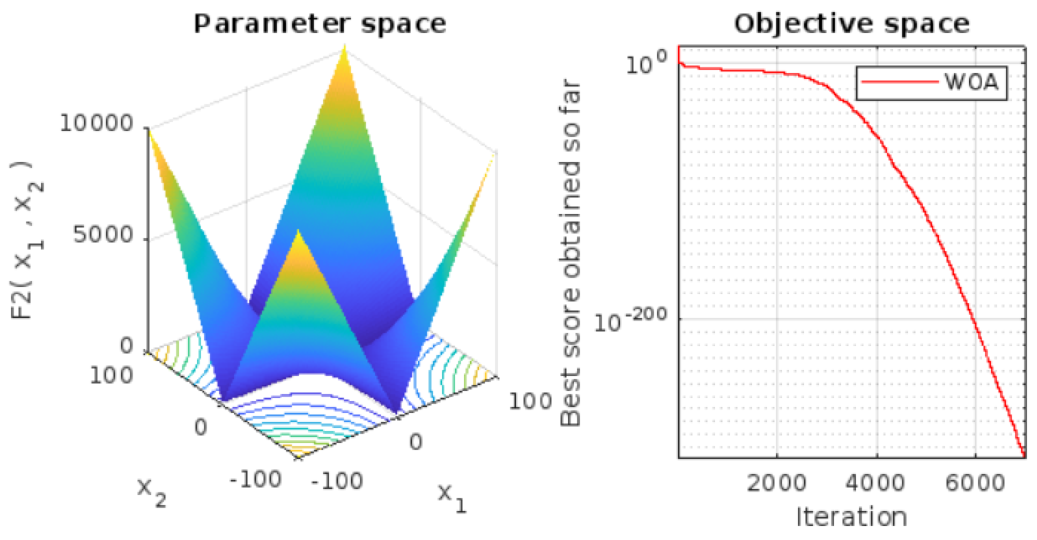
Rastrigin Function:

The best solution obtained by WOA is : [-0.28523 0.49495 0.51373 -0.89106 0.059453 -0.34192 0.92694 -0.48011 -0.0031175 -0.1608 0.71964 -0.09049 -0.69847 -0.25694 0.88842 -0.51051 -0.3259 0.86981 -0.92655 0.23062 0.49252 -0.60684 0.48435 0.06556 -0.36657 0.2408 0.045164 -0.084654 0.67011 -1.0221]
The best optimal value of the objective function found by WOA is : 3.6907



Rosenbrock Function:

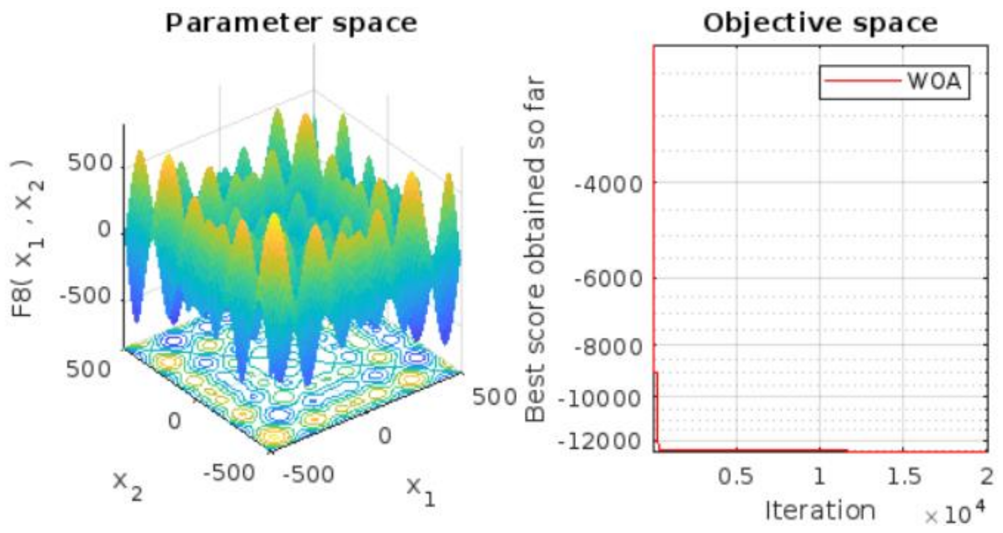
The best solution obtained by WOA is : [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
The best optimal value of the objective function found by WOA is : 0



Schwefel Function:

The best solution obtained by WOA is : [420.9732 420.9736 420.9499 420.9467 420.9772 420.9724 420.9475 420.9175 420.9723 420.9732 420.9694 420.9727 420.9737 420.9722 420.9779 420.9774 420.9736 420.9769 420.9719 420.9762 420.974 420.9749 420.9732 420.9747 420.9717 420.9734 420.9724 420.9564 420.9743 420.9721]

The best optimal value of the objective function found by WOA is : -12569.486



6. Genetic Algorithm(GA):

Ackley Function:

bestsol =[-1.6770   -1.7953   0.0231   -1.3658   -2.4309]

bestfitness = 7.512540777902085

Rastrigin Function:

bestfitness =53.3047
bestsol =-3.1021   -1.0112   -0.5074   -1.9463   -4.0050

Rosenbrock Function:

bestsol = [-1.7396   1.6996   2.9453   2.0589   3.3857]

bestfitness = 4.639269907340311e+03

Schwefel Function:

bestsol =[2.158363787463901e+02,-1.482802736e+02, 3.7571353501961e+02,2.979439587101e+02,3.947955564e+02]]

bestfitness = 1.029657055304744e+03

7. Gravitational Search Algorithm(GSA):

Ackley Function:

Mean of the global minimum values: 3.8749e-05

Standard deviation of the global minimum values: 3.4071e-09

Fbest = 3.6782e-04

std_deviation = 3.2841e-08

Rastrigin Function:

Mean of the global minimum values: 18.6972

Standard deviation of the global minimum values: 4.3644

Fbest = 19.8869

std_deviation = 4.3785

Rosenbrock Function:

Mean of the global minimum values: 4.9987

Standard deviation of the global minimum values: 0.48858

Fbest = 4.5427

std_deviation = 0.4891

Schwefel Function:

Mean of the global minimum values: -2384.257

Standard deviation of the global minimum values: 449.7658

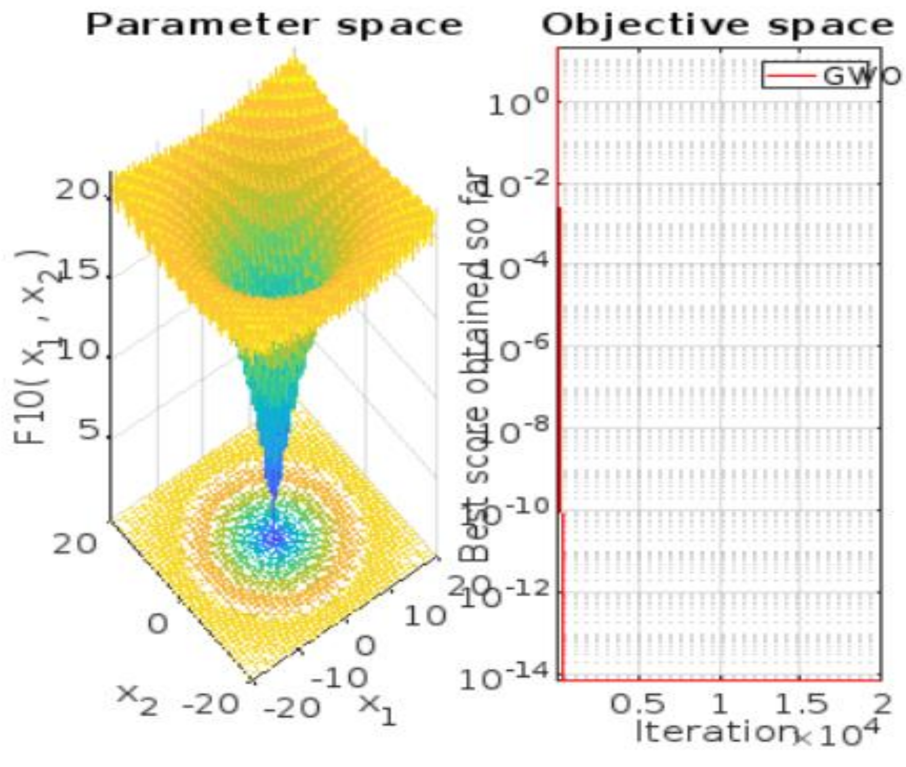
Fbest = -1.9837e+05

std_deviation = 450.4745

8. Grey Wolf Optimizer (GWO):

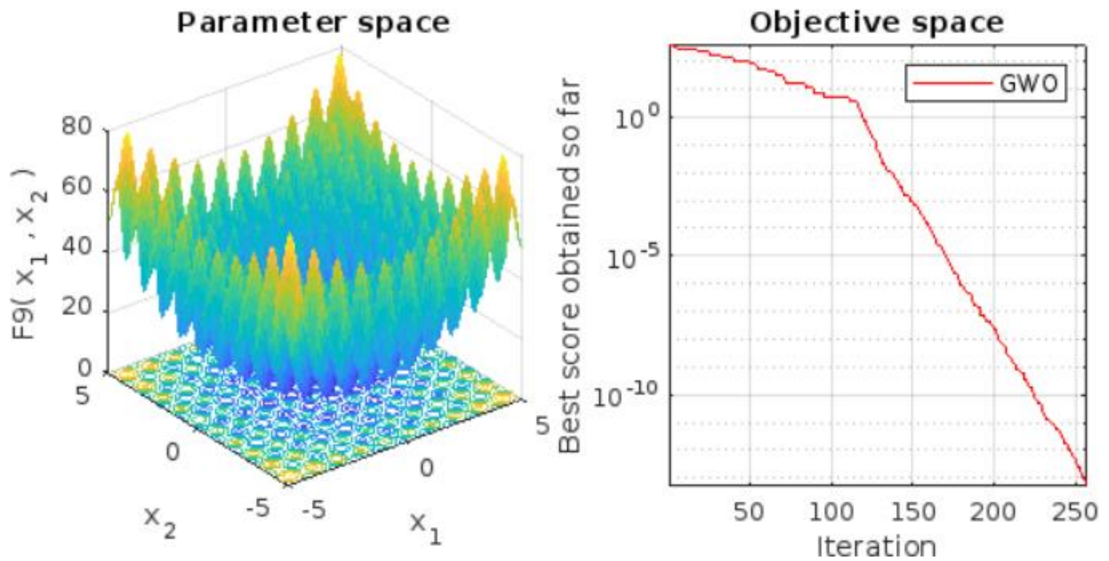
Ackley Function:

The best solution obtained by GWO is : -7.6642e-16 8.1368e-16 7.3422e-16 2.3918e-15 4.8398e-15 2.8673e-15 5.0263e-15 2.5511e-15 6.7032e-16 -2.3051e-16 -2.3266e-15 -3.94e-16 -2.1028e-15 7.9682e-16 -5.1017e-16 2.2274e-15 -4.4838e-17 -2.3556e-15 6.3068e-15 -5.7527e-18 2.4964e-15 1.5001e-15 -3.2982e-15 -2.9366e-15 6.7628e-17 -1.4375e-16 1.0639e-16 -1.7562e-15 -1.7362e-15 3.5635e-15
The best optimal value of the objective function found by GWO is : 7.5495e-15



Rastrigin Function:

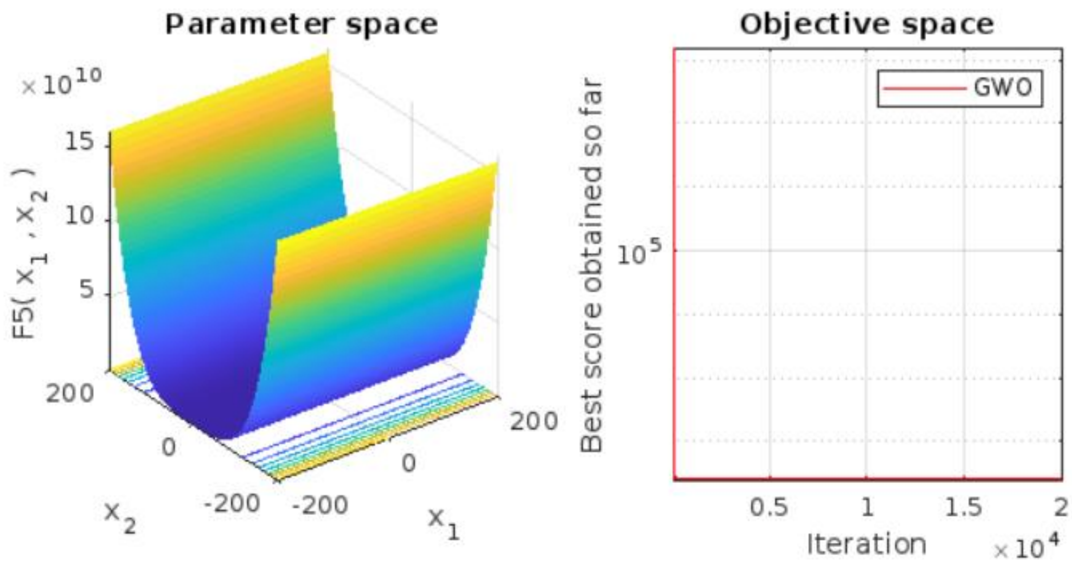
The best solution obtained by GWO is : 6.7419e-10 -1.9596e-09 -3.5457e-09   2.432e-10 -8.4738e-09 -9.3331e-10 -5.1682e-09 -6.8073e-10 7.9757e-09 -2.1125e-09 -5.0734e-09 3.4623e-09 3.379e-09 5.2535e-10 7.1592e-09 4.8042e-09 -3.5349e-09 -5.9844e-09   3.684e-09 -5.9586e-09 7.7694e-09 -1.2056e-09 7.1263e-09 -5.4675e-09 -1.7107e-09 -1.338e-09 -3.0272e-09 -1.044e-09 9.1068e-10 -2.7005e-09
The best optimal value of the objective function found by GWO is : 0



Rosenbrock Function:

The best solution obtained by GWO is : 0.91021   0.82796 0.68425   0.46492   0.20692   0.0010744 0.00017541 0.012423 5.6627e-06 -8.1149e-05 -1.7262e-06   0.0021469 0.001681 5.1734e-06   0.01013   5.92e-08   0.016354 0.0017331 0.00084955   -0.005328   0.0028438 -0.00020609 0.00039686 -0.00062619 -0.0014689   0.0047138   0.0053825 -0.0020235 6.6772e-05 -0.00014653

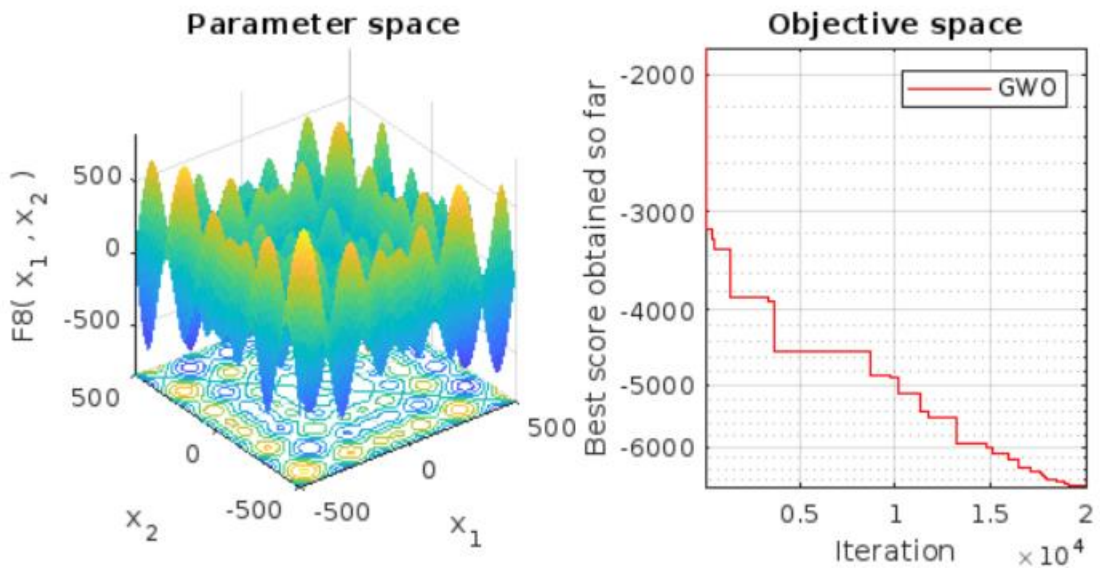
The best optimal value of the objective function found by GWO is : 25.1997



Schwefel Function:

The best solution obtained by GWO is : 420.9378 -28.47513 -26.67076 203.941 -302.4943 -9.900017 420.8962 -302.6054 421.0439 -27.42375 421.1023 1.341846 -2.601132 65.63145 65.69903 -302.6441 -3.758034 420.8843 -500 420.9548 -125.0737 420.9242 -302.5247 420.9939 -29.24989 420.9488 420.8168 -124.9254 203.8028 -302.4841

The best optimal value of the objective function found by GWO is : -6738.2869



9. Backtracking Search Optimization Algorithm (BSA):

Ackley Function:

global_minimum = 7.549516567451064e-15

Rastrigin Function:

global_minimum = 0

Rosenbrock Function:

global_minimum = 2.833489763940722e-28

Discussion:

A statistical comparison was conducted between the results obtained from the Differential Evolution Algorithm and MATLAB-implemented algorithms. Evaluation considered objective values and computational time, providing insights into the performance and efficiency of the DEA in comparison to other methods.

Both MATLAB and Python results for the Ackley function appear similar in terms of the best and worst costs obtained

The best minimum value for Rastrigin differs slightly between MATLAB and Python

There's a notable difference in the best minimum found for the Rosenbrock function

The results for the Schwefel function show differences in the best minimum value and the average minimum obtained.

Grey Wolf Optimizer (GWO) achieved similar or equivalent best objective values for the given benchmark functions compared to the results obtained from Differential Evolution (MATLAB) for most cases.

Comparing DE and GSA, we can observe differences in the mean and standard deviation of global minimum values and the best objective values obtained for each function. In some cases, GSA shows competitive results with DE, while in others, the performance differs notably. The choice of the algorithm could depend on the specific problem characteristics, convergence speed, and the requirement for global or local optima.

The comparison shows differences in the best fitness values obtained by DE and GA for each function. In some cases, GA performs competitively or better than DE, while in others, DE might achieve better results. These differences could stem from the specific characteristics of the optimization problem and the performance of each algorithm in navigating the search space to find the global or local optima. The choice between DE and GA could depend on factors like convergence speed, solution quality, and the nature of the problem being addressed.

The comparison indicates differences in the best optimal values obtained by DE and WOA for each function. WOA seems to find slightly different optima compared to DE for the given benchmark functions.

Conclusion:

In summary, our dedicated implementation of the Differential Evolution (DE) algorithm, as part of the assigned meta-heuristic algorithms, has provided valuable insights into its effectiveness across diverse benchmark functions. Through our independent development using a language other than Matlab, we gained a unique perspective on DE's adaptability and efficiency. The comprehensive comparison of our algorithm's results with those derived from Matlab implementations of all other algorithms, including DE itself, offers a nuanced understanding of its relative performance. This comparative analysis not only validates the efficacy of our implemented DE code but also contributes to a holistic evaluation of its competitiveness within the broader landscape of meta-heuristic optimization techniques. As we move forward, this research lays the foundation for further exploration, encouraging future investigations into parameter optimization strategies and the algorithm's applicability across a wider spectrum of optimization challenges.

References

1. Storn R and Price K, “Differential Evolution — a Simple and Efficient Heuristic for Global Optimization
2. Miguel Leon and N.Xiong - Investigation of Mutation Strategies in Differential Evolution for Solving Global Optimization Problems