Exploring the Multi-Verse Optimizer Algorithm with Various Optimization Functions

Yiran Liang, Xinyu Liu, Sally Sun, Wenlu Yu, Ziqian Wang

Abstract—Many popular optimization algorithms take inspiration from nature, such as Artificial Neural Networks (ANN), which mimic the function of human biological neurons, or Swarm Algorithm (SA) which simulates the collective behaviour of social animals and insects. This paper aims to study the Multi-Verse Optimizer (MVO) algorithm, proposed by Multi-Verse Optimizer: a nature-inspired algorithm for global optimization (Seyedali et al), which is motivated by the concepts of the multiverse theory. The following paper discusses the performance of the MVO on unimodal, multimodal, convex, and nonconvex functions. With the characteristics of MVO, the team concludes that it performs much better on complex nonconvex problems with numerous local optima.

The project implementation can be accessed at here.

Index Terms—Nature-Inspired Algorithm, Optimization Functions, Global Optimum, Complex Search Space

1 Introduction

This paper aims to discover the concept and performance of the Multi-Verse Optimizer (MVO) algorithm proposed by Multi-Verse Optimizer: a nature-inspired algorithm for global optimization (Seyedali et al). MVO is a physics-inspired technique. It's motivated by the three main concepts of the multiverse theory: white holes, black holes, and wormholes. White hole and black hole tunnels tend to transport objects from universes based on the inflation rate to improve the overall iteration's average inflation rate. A wormhole is a randomly existing space–time tunnel that connects the universes, which enables matter to travel between universes in order to balance its density. By utilizing these concepts, MVO exploits the search spaces to discover the best universe [1].

Related work will be examined to explore the build and performance of the algorithm. The team will apply datasets mentioned in the reference paper to replicate the benchmark functions, including linear and nonlinear, for the team to study the optimization process of the algorithm. Additionally, the performance of the MVO will be compared to other similar nature-inspired algorithms, including Grey Wolf Optimizer (GWO), Particle Swarm Optimization (PSO), and Genetic Algorithm (GA). Specifically, experiments will be conducted on convex and nonconvex optimization functions to compare results. Overall, the experiment aims to provide an understanding of the optimization problems best suited for the MVO algorithm and offer generalizations on its characteristics.

The findings of this study will provide the team with a better comprehension of the MVO algorithm's performance and its suitability for solving different types of problems. Overall, this study will contribute to the growing body of research on nature-inspired optimization algorithms and provide valuable insights for future research.

2 Related Work

Seyedali, Seyed, and Abdolreza have proposed a novel nature-inspired Multi-verse Optimizer (MVO) [1]. It begins the optimization process with a group of random solutions, also known as candidate solutions' sets, and then refines them through multiple iterations. Some of the popular algorithms in this class include PSO, GA, ACO, ABC, and GSA. One of the primary benefits of population-based algorithms is that they enable information exchange among candidate solutions, which facilitates the handling of issues like local optima, optima isolation, deceptiveness, search space bias, and premature convergence in a more efficient and effective manner [1]. Additionally, population-based algorithms have a lower probability of getting trapped in local solutions compared to single-solution-based algorithms. However, these algorithms also have some drawbacks, such as requiring a higher number of function evaluations per iteration and being less straightforward in terms of implementation [1].

Focusing on a population-based algorithm, MVO divides the search process into two phases: exploration versus exploitation. For the exploration phase, the concepts of white holes and black holes are utilized. There are a few assumptions: White holes are more likely to be created in universes with high inflation rates. Black holes are more likely to appear in universes with low inflation rates so they have a higher probability of receiving objects from other universes. In contrast, the wormholes assist in exploiting the search spaces. The convergence of MVO algorithms can be guaranteed by emphasizing exploitation [1].

Essentially, MVO uses terminologies in cosmology to represent the terms optimization algorithm. The algorithm assumes that each solution can be represented by a universe, and each variable in the solution is an object in that universe [1]. In addition, the algorithm assigns each

solution/universe an inflation rate, which refers to the expansion rate of that universe, and it is proportional to the corresponding fitness function value of the solution. Step size, which we are familiar with, is represented by the travelling distance rate in this algorithm, which defines how fast we can converge to the optimal solution. It is negatively proportional to search accuracy.

This paper will analyze the result of replicating the MVO algorithm using an open-source optimization framework called EvoloPy. As far as we know, the EvoloPy framework [2] is the initial tool that implements the nature-inspired algorithms in Python, which can effectively tackle computationally intensive optimization functions featuring extremely high dimensions.

3 Dataset

3.1 Data Description

We have selected 2 different sets of datasets for our experiment, 13 challenging test problems that were benchmarked in the referenced paper [1], which consist of 7 unimodal functions, where there is only one global optimum without other local optima, these functions will be used to test the exploitation of the algorithms, and 6 multi-modal test functions, where there is a global optimum as well as multiple local optima that exponentially increases with the dimensions, which could be used to test the exploration of the algorithms. The test functions are shown in Table 1.

The other dataset is from the public GitHub repository [3] provided to us by Professor Khalil, and consists of a collection of common optimization test functions. Six complex functions were chosen to test the efficiency of MVO in more complex optimization problems that are convex and nonconvex. The 4 functions chosen are, Ackley, Fletcher, Rastringin and Eggholder, which can be seen in Figure 1.

TABLE 1
Unimodal Benchmark Functions

Function	Dim	Range
$F_1(x) = \sum_{i=1}^{n} x_i^2$	50	[−100, 100]
$F_2(x = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i$	50	[−10, 10]
$F_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j\right)^2$	50	[−100, 100]
$F_4(x) = \max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	50	[−100, 100]
$F_5(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	50	[−30, 30]
$F_6(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	50	[−100, 100]
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1)$	50	[−1.28, 1.28]

TABLE 2
Multi-Modal Benchmark Functions

Function	Dim	Range
$F_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	50	[−500, 500]]
$F_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	50	[−5.12, 5.12]
$F_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)$ - $\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	50	[−32, 32]
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	50	[−600, 600]
$F_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2 [1+10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i+1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	50	[−50, 50]
$F_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)]\right\}$ + $\sum_{i=1}^{n} u(x_i, 5, 100, 4)$	50	[−50, 50]

3.2 Data Preparation

The 13 test problems benchmarked in the previous paper [1] were already implemented and available in the Evolopy [2] repository as functions. The parameters of the functions were tweaked to 50 universes and 500 iterations to match the experiment conducted by the previous paper[1]. The 6 other more challenging functions were manually implemented into the benchmarks.py[our github] to match the format of Evolopy.

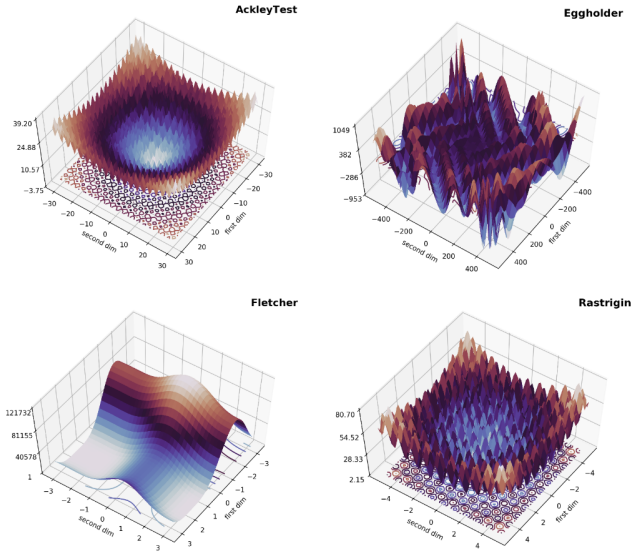

Figure 1. Convex & Nonconvex Optimization Functions

4 METHOD: MULTI-VERSE OPTIMIZER

4.1 Background

Multiverse Optimization (MVO) is a metaheuristic optimization algorithm inspired by the nature of multi-verse theory and big bang theory. The multiverse theory proposes that our universe is one of many universes that exist in parallel, each with its own unique physical laws and properties. MVO uses this concept to create a population of candidate solutions that exist in a multiverse of possibilities and evolve over time based on the fitness of each solution. It is based on a set of equations that simulate the behaviour of particles in a multiverse, including white holes, black holes, and wormholes, which are used to update the position of each particle in the population. The algorithm is able to handle multiple objectives and constraints and can be applied to a wide range of optimization problems.

4.2 MVO Algorithm

The inflation rate, which refers to the rate at which the universe is expanding, is pertinent when it comes to how the black holes and white holes in the MVO algorithm function. In fact, there are five rules that describe how the inflation rate affects the probability of different phenomena occurring in the context of Multi-Verse Optimization (MVO).

1. The higher the inflation rate, the higher probability of having a white hole: According to this rule, if the inflation rate is higher, there is a higher probability of having white holes in the universe. A white hole is a hypothetical object in space-time that acts as the reverse of a black hole, expelling matter and energy instead of drawing them in. This rule implies that higher inflation rates may result in more opportunities for matter and energy to escape through white holes.

2. The higher the inflation rate, the lower probability of having black holes: This rule suggests that if the inflation rate is high, the probability of having black holes is lower. A black hole is a region in space-time with an extremely strong gravitational field that nothing, not even light, can escape. The reasoning behind this rule is that higher inflation rates may prevent the formation of black holes due to the rapid expansion of space-time.

3. Universes with higher inflation rates tend to send objects through white holes: This rule suggests that in universes with higher inflation rates, there is a tendency for objects to be expelled through white holes. This may occur due to the increased probability of having white holes in these universes, as mentioned in the first rule.

4. Universes with lower inflation rates tend to receive more objects through black holes: This rule suggests that in universes with lower inflation rates, there is a tendency for objects to be drawn into black holes. This may occur due to the increased probability of having black holes in these universes, as mentioned in the second rule.

5. The objects in all universes may face random movement towards the best universe via wormholes regardless of the inflation rate: This rule suggests that all universes may have wormholes, which are hypothetical tunnels through space-time that could connect distant regions of the universe or even different universes. This rule implies that objects in any universe may be randomly moved through these wormholes, regardless of the inflation rate, in order to optimize the search for the best solution across all universes.

The process of optimization begins by generating a collection of arbitrary universes. During each iteration, objects within universes with high inflation rates have a tendency to move towards universes with lower inflation rates through the use of white or black holes. Additionally, all universes experience random teleportation of objects through wormholes to the best universe. Below are the pseudocodes for creating a white hole or a black hole, bounded by random variables normalized by the inflation rate.

```
SU=Sorted universes
NI=Normalize inflation rate (fitness) of the universes
for each universe indexed by i
        Black_hole_index=i;
        for each object indexed by j
                r1=random([0,1]);
                if r1<NI(Ui)
                        White_hole_index= RouletteWheelSelection(-NI);
                        U(Black_hole_index,j)= SU(White_hole_index,j);
                end if
        end for
end for
```

Figure 2. Pseudocodes for the Creation of White/Black Holes [1]

Based on five underlying assumptions, when a white or black tunnel is established between two universes, the universe with a higher inflation rate contains white holes, while the universe with a lower inflation rate is assumed to have black holes. The objects transfer from a high-inflation universe to a low-inflation universe, thereby ensuring that the overall average inflation rates of all universes improve over time. This process is repeated until convergence is achieved. Below are the pseudocodes for updating objects between universes through wormholes.

```
for each universe indexed by i
  for each object indexed by j
        r2=random([0,1]);
        if r2<Wormhole_existance_probability
                r3= random([0,1]);
                r4= random([0,1]);
                if r3<0.5
                        U(i,j)=Best_universe(j) + Travelling_distance_rate * (( ub(j) - lb(j)) *
                         r4 + lb(j));
                else
                        U(i,j)=Best_universe(j) - Travelling_distance_rate * (( ub(j) - lb(j)) *
                        r4 + lb(j));
                end if
        end if
        end for
end for
```

Figure 3. Pseudocodes for Updating Variables [1]

Figure 3 shows the pseudocodes for updating variables between each universe and the best universe. It can be

inferred that the key coefficients are the probability of wormhole existence (WEP) and the rate of travelling distance (TDR). WEP is used to determine the likelihood of a wormhole's presence in a given universe and must increase steadily during the optimization process to emphasize exploitation. On the other hand, TDR specifies the maximum distance that an object can be teleported by a wormhole around the most optimal universe found so far, which is similar to the step size in optimization problems. TDR decreases over iterations to allow for more precise exploitation and local search. Unlike WEP, TDR does not need to increase linearly.

Putting together, the existence probability of wormholes in universes can be smoothly increased by increasing adaptive WEP values, which serves to emphasize exploitation during the optimization process. By contrast, decreasing TDR values can reduce the travelling distance of variables/objects around the most optimal universe, resulting in a more accurate local search over successive iterations. The formula for both WEP and TDR are:

$$\text{WEP} = \min + l \times \left(\frac{\max - \min}{L}\right)$$

where l is the current iteration, and L indicates the maximum iterations.

$$\text{TDR} = 1 - \frac{l^{1/p}}{L^{1/p}}$$

where p is referred to as the exploitation accuracy over the iterations. The higher p, the sooner and more accurate exploitation/local search.

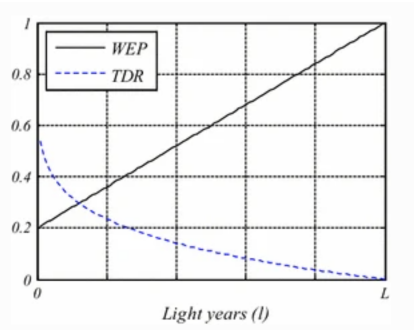

Figure 4. WEP & TDR Values Over Iterations [1]

In order to determine the computational complexity of the proposed algorithms, multiple factors are considered, such as the number of universes, iterations, and the specific roulette wheel and universe sorting mechanisms utilized. Sorting the universe with Quicksort is implemented in each iteration, possessing O(n log n) as the best possible complexity and $O(n^2)$ as the worst-case scenario. In contrast, the roulette wheel selection is executed for each variable in every universe during iterations. Its complexity depends on the specific implementation adopted, which can either be O(n) or O(log n). To summarize, the overall computational complexity is:

$$O\left(\text{MVO}\right) = O\left(l\left(O\left(\text{Quick sort}\right) + n \times d \times \left(O\left(\text{roulette wheel}\right)\right)\right)\right)$$

$$O\left(\text{MVO}\right) = O\left(l\left(n^2 + n \times d \times \log n\right)\right)$$

where n = universe number, l = max. iteration number, and d = object number.

5 Experiments

5.1 Experiment Setup

The original experiment was conducted using Matlab but our experiments were implemented using PyCharm CE, and all the Python code can be found here[hyperlink to github]. We used Python because of its simple and clear syntax, open-source scientific computing libraries, and packages. The experiment was all conducted on our own computers, running on an Intel 8700k CPU. Existing code from Evolopy [2] was used for the implementation of all the optimization functions as well as the 13 challenging functions. The additional functions for the more complex convex and nonconvex optimization functions were transformed from Pasa Opasen's GitHub repository [3] to fit Evolopy's module structure code. Numpy and sklearn were used in constructing the search space as well as other mathematical operations. Pandas and matplotlib were used to visualize the results.

To match the experiment difficulty that was tested by S. Mirjalili [1], the optima of test functions 1-13 were randomly shifted every run. To balance out the lucky runs since there is a randomness to stochastic optimizers, the average score of 30 runs is compared against each other. The number of iterations was set to 500, and the number of universes was 30, with WEP increasing linearly from 0.2 to 1 and TDR decreasing from 0.6 to 0 to match S. Mirjalili's [1] experiment. MVO was matched against PSO, one of the best algorithms among SI-based techniques, GA as the best evolutionary algorithm, and GWO as one of the best overall algorithms, to validate the efficiency and accuracy of the results. The results were all compared with S. Mirjalili's paper [1] for cross-validation.

The performance metrics measured in the experiments were the convergence of each algorithm over 500 iterations, the fitness level, and the execution time per run. The main metric we want to look at is the convergence comparison since this determines how fast and accurately MVO reaches the global optimum among the others. Execution times were also recorded since for optimization algorithms to be actually feasible in real life, they must be able to solve complex search spaces in a reasonable time.

5.2 Experiment Results & Discussion

The experiment results were stored in the results folder here, consisting of the convergence graph and the experiment averages.

5.2.1 Unimodal Results

The unimodal test functions 1-7 were to indicate the exploitation ability of the optimization function. The results in Table 3, show that MVO was only able to outperform other algorithms in a few cases as stated in the previous paper [1]. This suggests that MVO has a decent exploitation ability, which is the due to the integration of the adaptive WEP and

TABLE 3
Experimental Results

F	MVO		GWO		PSO		GA	
	Exe.	Fit.	Exe.	Fit.	Exe.	Fit.	Exe.	Fit.
1	0.1	10.2	0.1	0	0.1	0.21	0.1	14522.9
2	0.1	273.9	0.1	0	0.1	25.96	0.1	68.23
3	0.1	5865	0.1	3.1	0.1	1587.4	0.1	46178.65
4	0.1	16.5	0.1	0	0.1	3.99	0.1	62.36
5	0.1	733.4	0.1	47.5	0.1	418.23	0.1	13879083
6	0.1	10.2	0.1	2.4	0.1	0.15	0.1	14876
7	0.1	0.1	0.1	0	17.5	0.1	0.5	13.1
8	0.8	-12057	0.3	-8501	0.3	-7371.1	0.3	-10563.1
9	0.6	248.1	0.9	15.7	0.9	285.57	0.9	253.3
10	0.5	3.5	0.5	0	0.5	1.56	0.5	16.65
11	0.8	1.1	0.3	0	0.3	0.01	0.3	127.4
12	0.6	6	0.9	0.1	0.9	0.1	0.9	6153439
13	0.6	13.5	0.9	2.1	0.9	0.24	0.9	32860619

TDR constants to balance the exploration and exploitation of the algorithm, encouraging exploitation when the inflation rate is low and the fitness level is low. But in some cases, as seen in Table 3, MVO did not perform better than other algorithms, in fact, it was extremely far from the global optima, which is contradicting the previous findings[1]. We hypothesize this could be due to the randomness of stochastic algorithms, but also since MVO does prioritize exploration at the beginning, it converges slower during the first few iterations.

5.2.2 *Multi-Modal Results*

The population-based stochastic algorithm is designed to solve more complex optimization problems where there are multiple local optima, so it was expected for MVO, PSO, and GWO to outperform GA. The results in Appendix B, from functions 8 to 13 indicate the exploration ability of MVO, as it was converge every iteration not stuck to any.

The exploration ability of MVO is due to the existence of white holes and black holes, as objects travel from one universe to the other, it causes sudden changes to the fitness level and promotes exploration. This avoids the problem of local optima stagnation, as black holes are created in stagnated universes to increase exploration through increasing the inflation of the universe. The only graph in that we were able to replicate the same result was Function 8 in the graph on the right in Figure 5, and it is the only time where MVO was able to outperform.

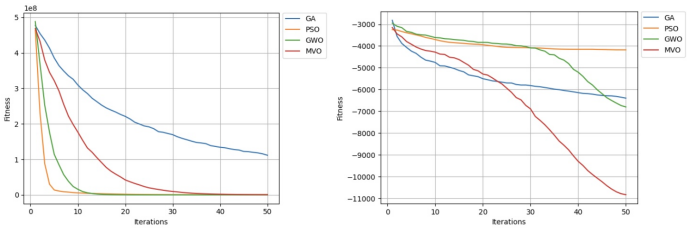

Figure 5. Convergence Graph for Functions 5 & 8

5.2.3 *Convex & Nonconvex Optimization Functions*

This subsection discusses the balance between exploration and exploitation, as the nonconvex optimization functions can be extremely challenging with all the rough surfaces and local optimums. The convergence graph can be seen in Appendix D. The results showed that MVO was able to greatly outperform other algorithms when the exploration space become more complex, as seen in the Eggholder function in Figure 1, where there are numerous amounts of local optima and the global optimum expanding outward, MVO was able to continuously converge over the 500 iterations (Figure 6) as others were either converging much slower or were stuck within a local optimum.

This is credited to the use of the adaptive WEP, which helps to balance the exploration and exploitation of each universe.

Contrary to the findings in the 13 functions, where MVO didn't outperform others, in the 4 additional convex and nonconvex functions, MVO was able to generally outperform. We could conclude that MVO does better with more complex problems.

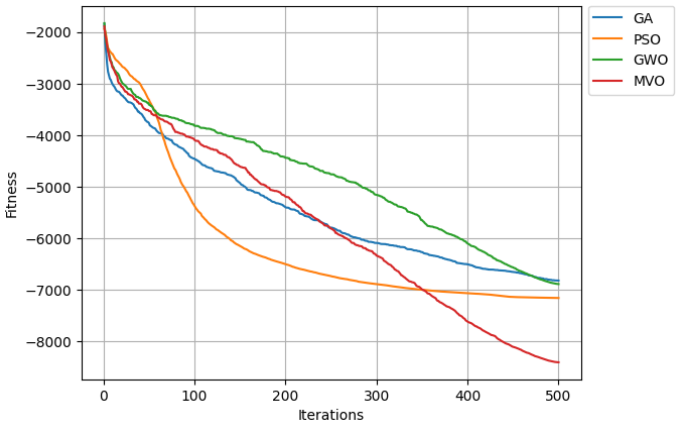

Figure 6. Eggholder Convergence with 500 Iteration & 50 Population

5.2.4 *Convergence Analysis*

To visualize the convergence of the proposed algorithms, we provided the converges graph of each function in Appendix B. From the convergence graph, we can see that MVO is able to improve consistently through each iteration, but it is not the fastest converging algorithm at the beginning, as it is a more balanced algorithm for exploration and exploitation.

As the results show, MVO is capable of always converging towards the global optimal as iteration increases, which can be reasoned by the concept of white and black holes, where variables of universes will travel from a fit universe to a less fit universe to help with exploration, so when a local optimum is reached, the variables will travel to other universes to emphasis exploration instead of continuously exploiting the local optima with many variables.

Surprisingly, in the Ackley function, with numerous rough surfaces, GA exhibited the most consistent improvements. For other nonconvex functions, especially the Eggholder function, MVO was significantly better than the other algorithms as the iteration increased as seen in Figure 6.

5.2.5 *Changing Parameters*

The team also looks at different sets of limitations to range, iteration, and populations. As seen in Figure 7, with a smaller population size of 30 and iteration set to 50, MVO converged much faster compared to other algorithms. But looking at the original convergence graph in Figure 5, PSO was similar until somewhere past 50 iterations where it made a huge jump quickly passing other algorithms. This could be due to the randomness of the algorithms and search variables, but the general trend of MVO has the most consistent improvement over each iteration.

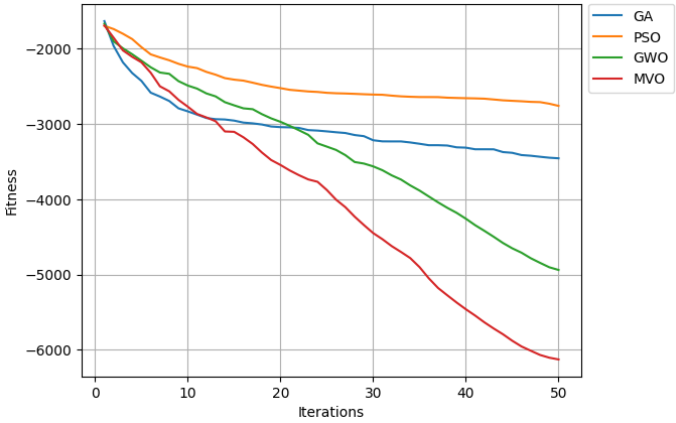

Figure 7. Eggholder Convergence with 50 Iterations & 30 Population

6 Conclusion

In conclusion, our project aimed to implement the multiverse optimizer (MVO) algorithm in Python and compare its performance with other optimization algorithms. We conducted experiments on benchmark functions, both unimodal and multimodal, and found that while MVO outperformed other algorithms in some functions, it did not consistently outperform other stochastic population-based algorithms. We also tested the algorithms on common optimization functions and found that all population-based stochastic algorithms performed comparably well, with GA failing to find the global optimum. We characterize that MVO performs better on more complex problems, like the nonconvex functions tested in the paper, as MVO is able to steadily improve every iteration despite not being the fastest exploiting algorithm. The team also found that when limiting the population to a smaller amount, MVO was able to converge much faster compare to other algorithms, and showed superior overall performance. To summarize, MVO is suited for complex problems, with limited variables.

7 Future Work

In the future, our team aims to enhance the MVO algorithm's performance by testing it on more complex and real-world optimization problems. This includes exploring machine learning models and incorporating additional data features to improve the algorithm's accuracy and robustness. We also plan to investigate the sensitivity of the MVO algorithm to its parameters and determine the optimal settings to maximize its performance for specific problems. Moreover, we recognize the importance of integrating current research in the field to ensure our project is up-to-date with the latest advancements. To achieve this, we intend to compare MVO with other state-of-the-art optimization algorithms on a wider range of benchmark functions and real-world problems. This may involve examining recent studies or consulting with experts to expand our knowledge and potentially contribute to the field by building upon or expanding on current research.