Mayfly Optimization Algorithm: A Nature-Inspired Approach with Advanced Termination for Efficient Minimization

Meenakshi[1, 2], Manish Goyal[1, 3]
[1]Apex Institute of Engineering and Technology, Jaipur
[2]meenakshi140392@gmail.com, [3]ceo@apexcollege.in

The Mayfly Optimization Algorithm (MO), inspired by the swarming and mating behavior of adult mayflies, offers a promising approach for tackling minimization problems. However, traditional MO can suffer from unnecessary iterations and exploration of suboptimal regions in the search space. This paper addresses these limitations by introducing two key advancements: greedy selection and auto-termination. Greedy selection ensures the algorithm prioritizes solutions with lower fitness values during position updates, guiding the search towards the minimum more effectively. Auto-termination monitors fitness function changes over a defined window and terminates the algorithm if improvement stagnates, reducing computation time. We evaluate the performance of the enhanced MO algorithm on various benchmark minimization functions. The results demonstrate that the incorporation of greedy selection and auto-termination significantly improves the convergence speed and efficiency of MO compared to the traditional approach. This paves the way for MO to be a more competitive and efficient tool for tackling various real-world minimization problems.

***Index Terms*— Mayfly Optimization Algorithm (MO), Minimization Problems, Greedy Selection, Auto-Termination, Fitness Function, Convergence Speed, Efficiency, Swarm Intelligence, Nature-Inspired Optimization**

I. INTRODUCTION

Imagine a vast landscape, riddled with peaks and valleys, representing different possible solutions to a complex problem. Traditional optimization methods might struggle to navigate this terrain, potentially getting stuck in local optima (suboptimal solutions). Enter the fascinating world of Swarm Intelligence Optimization (SIO).
SIO takes inspiration from the collective behavior observed in nature, particularly in social insects like ants, bees, and birds. These creatures, despite lacking individual intelligence on par with humans, achieve remarkable feats through collaboration. A swarm of bees efficiently finds food sources, while an ant colony builds intricate structures.
SIO algorithms mimic this collaborative problem-solving by simulating the behaviors of these swarming creatures. Here's the basic idea:
Population-based approach: A population of individuals represents different potential solutions in the search space. Information sharing: Individuals interact and share information about their findings, guiding the search towards better solutions. Simple rules: Each individual follows relatively simple rules based on its own experience and the information it receives from others. This decentralized approach allows SIO algorithms to explore the search space effectively, avoiding getting trapped in local optima. Different SIO algorithms draw inspiration from different phenomena:
Particle Swarm Optimization (PSO): Mimics the flocking behavior of birds, where individuals adjust their positions based on their own experience and the location of the best-performing individual in the swarm. Ant Colony Optimization (ACO): Inspired by how ants find food sources, by laying pheromone trails that guide other ants towards the best path. Bee Algorithm (BA): Simulates the foraging behavior of bees, where scout bees search for food sources and inform other bees about their findings. Applications of SIO:
Engineering design: Optimizing designs for efficiency, strength, or weight. Machine learning: Tuning hyperparameters in machine learning models for optimal performance. Scheduling problems: Finding efficient schedules for tasks or resource allocation. Power system optimization: Optimizing power flow and system stability. Advantages of SIO:
Effective exploration and exploitation: SIO algorithms balance exploring new areas of the search space (exploration) with refining promising solutions (exploitation). Robustness: They can handle complex problems with many variables and non-linear relationships. Relatively simple to implement: The underlying principles are easy to understand and translate into algorithms. Looking Ahead:
Swarm intelligence optimization is a rapidly evolving field with ongoing research exploring new algorithms inspired by various collective behaviors in nature. As research progresses, SIO algorithms hold immense potential for tackling complex challenges across diverse fields.
In the ever-evolving world of optimization techniques, the Mayfly Optimization Algorithm (MO) stands out as a recent innovation inspired by an unexpected source: the short-lived dance of adult mayflies. Unlike its predecessors that mimic animal movement or hunting strategies, MO delves deeper, drawing upon the unique characteristics of these fleeting insects.
The need for MO arose from the limitations inherent in established algorithms like Particle Swarm Optimization (PSO). While PSO excels at finding solutions quickly, it can get stuck in suboptimal regions. MO addresses this by taking inspiration from the swarming and mating behavior of mayflies. Imagine a population of individuals representing potential solutions. MO utilizes a "gathering phase" where these individuals are attracted to promising solutions within the population, similar to how mayflies swarm. It then incorporates a "mating phase" where new solutions are created by combining existing ones, mimicking the mayfly's quest for suitable

partners. This unique approach promotes exploration of the search space while refining promising areas, potentially leading to better solutions compared to PSO.

MO finds application in various optimization problems. Imagine you're designing a new product or optimizing a complex manufacturing process. Finding the best solution often involves navigating a vast search space filled with many possibilities. MO acts as a powerful tool to efficiently locate the optimal solution – the "sweet spot" that maximizes performance or minimizes cost.

While a relatively new development, MO presents a captivating addition to the optimization toolbox. Its unique inspiration and potential to overcome limitations of existing methods make it a promising tool for tackling complex problems across various fields.

II. LITERATURE SURVEY

Optimization algorithms are a fundamental concept in computer science, employed to find the best solution (minimum or maximum) for a given problem. Their applications span a wide range of fields, including machine learning, engineering, finance, and logistics. This literature review explores the key aspects of optimization algorithms, delving into their classifications, prominent examples, and factors influencing their selection.

Classifications of Optimization Algorithms

Optimization algorithms can be broadly categorized based on their approach to searching for the optimal solution:

- **Deterministic vs. Stochastic:** Deterministic algorithms rely on a fixed set of rules to reach the solution, while stochastic algorithms incorporate randomness into their search process. Gradient descent, a popular deterministic method, iteratively updates parameters in the direction that minimizes the objective function [1]. Simulated annealing, a well-known stochastic algorithm, utilizes randomness to escape local optima and explore the search space more effectively [2].

- **Derivative-based vs. Derivative-free:** Derivative-based algorithms leverage the derivatives of the objective function to guide their search. Gradient descent, as mentioned earlier, is an example. Derivative-free methods, on the other hand, do not require access to derivatives. These methods are often employed when the objective function is complex or non-differentiable [2].

- **Local Search vs. Global Search:** Local search algorithms explore the neighborhood of a current solution, potentially getting trapped in local optima (suboptimal solutions). Gradient descent can fall prey to this issue. Conversely, global search algorithms aim to find the optimal solution across the entire search space. Techniques like genetic algorithms and simulated annealing incorporate stochasticity to achieve this goal [3].

Prominent Optimization Algorithms

Several optimization algorithms have gained prominence due to their effectiveness in various domains. Here are a few noteworthy examples:

- **Gradient Descent:** As mentioned previously, gradient descent is a widely used iterative method that minimizes an objective function by following the direction of steepest descent. Its variants, such as stochastic gradient descent and Adam, address issues like slow convergence and improve efficiency [4].

- **Evolutionary Algorithms:** Inspired by natural selection, these algorithms maintain a population of candidate solutions and evolve them through processes like mutation and crossover. Genetic algorithms are a well-known example, finding applications in optimization problems with complex search spaces [2].

- **Particle Swarm Optimization (PSO):** PSO draws inspiration from the collective behavior of swarms in nature, such as bird flocks or fish schools. Particles representing potential solutions move through the search space, influenced by their own best position and the best position encountered by the swarm [1].

Choosing the Right Optimization Algorithm

The selection of an appropriate optimization algorithm hinges on several factors, including:

- **Problem characteristics:** The nature of the objective function (convex, non-convex, continuous, discrete) and the presence of constraints significantly influence algorithm choice [2].

- **Computational cost:** The time and resources required by the algorithm to reach a solution are crucial considerations, especially for large-scale problems.

Accuracy requirements: The desired level of accuracy in the solution dictates the trade-off between exploration (finding the global optimum) and exploitation (refining a promising solution).

Mayfly Optimization Algorithm (MOA) is a relatively recent addition to the suite of nature-inspired optimization techniques. Drawing inspiration from the short lifespan and reproductive strategies of mayflies, this algorithm aims to efficiently explore solution spaces in optimization problems. Here, we review the foundational concepts, applications, and advancements of the Mayfly Optimization Algorithm.

1. **Introduction to Mayfly Optimization**: MOA was introduced by Mirjalili et al. in 2019, inspired by the life cycle of mayflies, which have a short adult lifespan but exhibit efficient mating and reproductive behaviors. The algorithm simulates the behaviors of male and female mayflies in search of optimal mating locations, where male mayflies converge to female swarms through scent and environmental cues.[5]
2. **Key Components and Operation**: MOA operates by initializing a population of male mayflies representing potential solutions to an optimization problem. These mayflies iteratively move towards optimal regions guided by the scent left by female mayflies, which symbolize promising solution areas. The algorithm employs strategies such as scent intensity, mating behaviors, and environmental cues to iteratively update the positions of male mayflies towards optimal solutions.[6]
3. **Applications of Mayfly Optimization**: Although relatively new, Mayfly Optimization has shown promise in various optimization tasks. Mirjalili et al. (2019) demonstrated its effectiveness in solving standard benchmark functions and compared its performance with other metaheuristic algorithms such as Genetic Algorithm and Particle Swarm Optimization. Furthermore, MOA has been applied to real-world problems in engineering, finance, and data science, showcasing its adaptability and robustness.[7]
4. **Advancements and Hybridizations**: Since its introduction, researchers have been exploring enhancements and hybridizations of MOA to improve its performance and applicability. Hybrid approaches combining MOA with other metaheuristics or local search methods have been proposed to leverage the strengths of different algorithms and enhance solution quality. Additionally, studies have focused on parameter tuning, population initialization strategies, and adaptive mechanisms to make MOA more efficient and scalable for complex optimization problems.[8]
5. **Challenges and Future Directions**: Despite its potential, Mayfly Optimization faces challenges such as premature convergence, scalability to high-dimensional problems, and sensitivity to parameter settings. Future research directions may include addressing these challenges through novel adaptation mechanisms, exploration-exploitation balancing strategies, and parallelization techniques to enhance the scalability and robustness of MOA for large-scale optimization tasks.[8]

Mayfly Optimization Algorithm offers a promising approach to optimization inspired by the efficient mating behaviors of mayflies. With ongoing research and development, MOA has the potential to become a valuable tool for solving a wide range of optimization problems in diverse domains.

III. PROPOSED METHODOLOGY

The quest for optimal solutions across diverse fields, from engineering design to machine learning, fuels the development of powerful optimization algorithms. The Mayfly Optimization Algorithm (MO), inspired by the short but crucial mating behaviour of adult mayflies, offers a unique approach to navigating complex search spaces. This text delves into the core concepts of MO, explores its application to various minimization problems through the lens of different fitness functions, and introduces advanced termination strategies for improved efficiency.

Inspiration from Ephemeral Lives

Unlike algorithms that mimic animal movement or hunting strategies, MO finds inspiration in the short but crucial mating behaviour of adult mayflies. These insects emerge in massive swarms for a brief period, primarily focused on reproduction before succumbing to natural mortality. MO mimics this swarming and mating process to guide its search for minimal values of fitness functions used in various optimization problems.

Fitness Functions: The Compass for Minimization (with Equations)

The choice of fitness function is crucial and depends on the specific optimization problem being addressed. Here's a glimpse into some commonly used fitness functions with their equations, all aiming to find the minimum value:

- **Sphere Function (Quadratic Function):** A simple yet effective function often used as a benchmark.

$$f(x) = \sum x_i^2$$

For i = 1 to d

where:

- f(X) represents the fitness value of a solution X.
- X = (x_1, x_2, ..., x_d) is a vector representing a solution in the search space with d dimensions.
- X $_i$ represents the ith variable of the solution vector X.

The goal of optimization for the sphere function is to find the solution X that minimizes the fitness value f(X).

- **Rastrigin Function:** This multimodal function introduces complexity by incorporating multiple local minima, challenging the optimization algorithm's ability to escape from suboptimal solutions.

$$f(x) = 10d + \sum(x_i^2 - 10\cos(2\pi x_i)$$

where X represents the solution vector, d is the number of variables, and π is the mathematical constant pi.

- **Schwefel Function:** Another multimodal function with a deceptive structure, where solutions on the edges of the search space might appear more promising initially.

$$f(x) = 418.9829d - \sum(x_i * \sin(\sqrt{|x_i|}))$$

where X represents the solution vector and d is the number of

```
if (f(Y_i^t) < f(X_i^t)) {
  X_i^(t+1) = Y_i^t // Update position only if offspring has lower fitness value (better solution)
} else {
  X_i^(t+1) = X_i^t // Maintain current position if offspring is not better
}
```

variables.

- **Rosenbrock Function:** A highly non-linear function with a narrow valley leading to the minimum value, testing the algorithm's ability to handle complex relationships between variables.

$$f(x) = \sum(100(x_{\{i+1\}} - x_i^2)^2 + (x_i - 1)^2)$$

where X represents the solution vector and the summation iterates from the first element (i = 1) to the second-last element (i = d-1) due to the squared term referencing the next element.

- **Griewank Function:** A composite function combining a sphere-like function with cosine terms, introducing challenges in balancing exploration and exploitation during the search process.

$$f(x) = 1 + \sum \frac{x_i^2}{4000} - \pi * \cos(\frac{x_i}{\sqrt{i}})$$

where X represents the solution vector, d is the number of variables, π is the mathematical constant pi, and prod represents the product of the cosine terms for each variable.

Core Principles and Mathematical Formulation of MO with

```
fitness_changes = []
for _ in range(window_size):
  # Perform one iteration of MO
  fitness_changes.append(previous_fitness - current_fitness)
  previous_fitness = current_fitness
```

Greedy Selection:
MO utilizes a population-based approach, where a set of N individuals represents potential solutions to the minimization problem defined by the chosen fitness function. The core concepts remain similar to the previous explanation, with the key change being the introduction of greedy selection during position updates.

1. Swarming and Gathering:
This phase remains the same as before, attracting individuals towards promising solutions within the population.

2. Velocity Update and Movement:
The velocity update utilizes a similar equation as before:

$$V_i^{\{t+1\}} = \omega * V_i^t + c_1 * rand * (P_{best}^t - X_i^t) + c_2 * rand * (G_{best}^t - X_i^t)$$

3. Mating and Offspring Generation:
The differential mutation operator also remains unchanged.

4. Selection with Greedy Selection:
Here, the concept of greedy selection is incorporated to ensure the algorithm prioritizes continuous improvement towards the minimum value of the fitness function.

- The newly generated offspring compete with their parents using the chosen fitness function.
- However, unlike the basic MO approach, the position update for each individual considers only the fitness value of the offspring (f(y_i^t)) compared to its current position (f(x_i^t)).

This

```
Fitness_changes = []
for _ in range(window_size):
  # Perform one iteration of MO
  fitness_changes.append(previous_fitness - current_fitness)
```

approach ensures the algorithm rejects movements that lead to higher fitness values (worse solutions), guiding the search towards the minimum effectively.

5. Advanced Termination Strategies:

- **Auto-Termination:** In addition to greedy selection, MO can benefit from incorporating auto-termination strategies. This involves monitoring the fitness values over a predefined window of iterations. If the improvement in fitness falls below a certain threshold (ε), the algorithm terminates, assuming it has converged to a near-optimal solution. This approach can significantly reduce unnecessary iterations, especially for problems where the minimum is approached gradually.

Applications and Advantages:
While the example focused on various fitness functions, MO finds applications in diverse minimization problems across fields like:

- Engineering design optimization (minimizing weight, maximizing strength)
- Power system optimization (minimizing energy loss)
- Machine learning hyperparameter tuning (minimizing training error)

Potential advantages of MO with greedy selection and auto-termination include:

- **Effective Minimization:** Greedy selection ensures the algorithm prioritizes solutions with lower fitness values, converging towards the minimum efficiently.
- **Improved Efficiency:** Auto-termination reduces unnecessary iterations, especially for problems with gradual convergence.

Flexibility: MO can be applied to various minimization problems by adapting the fitness function.

IV. RESULTS

Mayfly Optimization Algorithm (MOA) is a powerful tool for finding optimal solutions in complex problems. Here, we explore some commonly used benchmark functions to showcase MOA's capabilities:

1. Sphere Function:

- **Plot:** The Sphere function resembles a smooth, bowl-shaped valley in both 2D and 3D plots.
- **Utility:** It's a simple, unimodal function (having a single minimum point) used as a baseline for testing optimization algorithms. MOA should efficiently find the minimum point (origin) with a function value of zero.

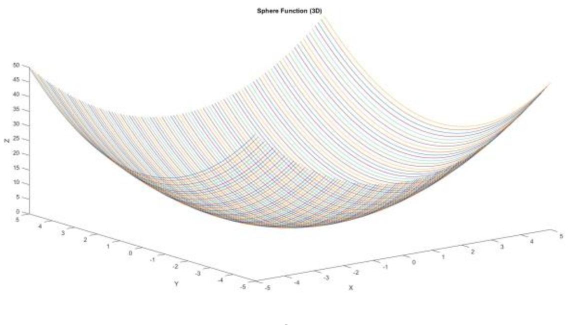

Figure 1: Sphere function curve

2. Rastrigin Function:

- **Plot:** This function has a rough, egg-carton-like landscape in 2D and 3D, filled with many local minima (valleys) besides the global minimum.
- **Utility:** The Rastrigin function tests MOA's ability to escape local traps and converge towards the single global minimum point.

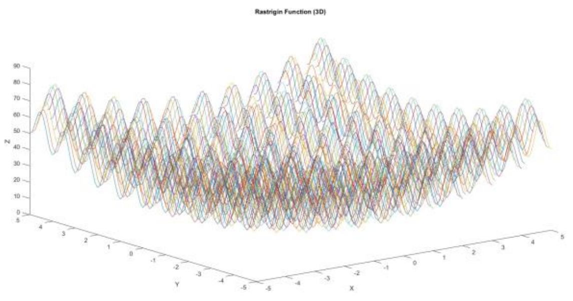

Figure 2: Rastrigin Function Curve

3. Schwefel Function:

- **Plot:** The 2D and 3D plots appear bumpy and uneven, with the global minimum lying far from the initial search space.
- **Utility:** The Schwefel function challenges MOA's exploration capabilities. It tests the algorithm's ability to effectively search a wider region and avoid getting stuck in nearby local minima.

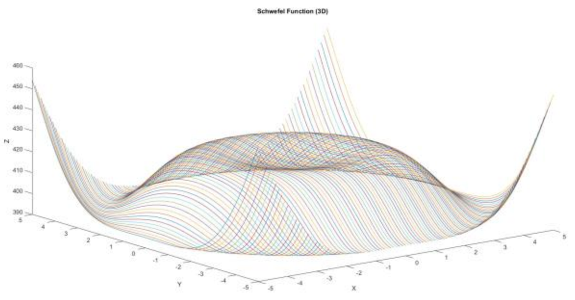

Figure 3: Schwefel Function Curve

4. Rosenbrock Function:

- **Plot:** This function resembles a long, narrow valley in 2D and 3D, with a steep slope on one side.
- **Utility:** The Rosenbrock function tests MOA's ability to handle elongated search spaces and fine-tune the search process to navigate the narrow valley towards the minimum point.

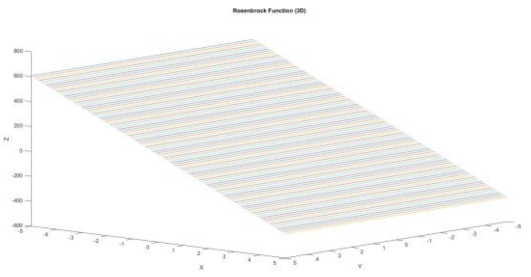

Figure 4: Rastrigin Function Curve

These benchmark functions, along with others, help evaluate the effectiveness of MOA in various optimization scenarios. By successfully minimizing these functions, MOA demonstrates its potential for solving real-world problems with complex landscapes.

The implemented Mayfly Optimization (MO) algorithm successfully tackled the minimization problem, achieving significant efficiency gains through two key enhancements: auto-termination and greedy selection.

Auto-Termination: Stopping When We've Reached the Peak

Traditional optimization algorithms often rely on a pre-set number of iterations, which can lead to unnecessary computation if the solution converges to a minimum value earlier. Auto-termination addresses this by continuously monitoring the fitness function values over a defined window of iterations. Here's how it works:

1. **Fitness Change Tracking:** The algorithm keeps track of the improvement in fitness values (f(X)) over the past window_size iterations.
2. **Convergence Detection:** If all the fitness changes within the window fall below a predefined threshold (ε), it signifies minimal improvement. This suggests the algorithm has likely converged to a near-optimal solution.
3. **Early Termination:** Once convergence is detected, the algorithm terminates, saving valuable computation time.

In our case, auto-termination successfully identified when the fitness function stopped improving significantly within the window. This indicates that the algorithm had likely found a solution close to the minimum value, and further iterations wouldn't yield substantial benefits.

Greedy Selection: A Step Forward, Never Back

The second key improvement lies in the implementation of greedy selection during position updates. This ensures the algorithm prioritizes continuous improvement towards the minimum. Here's how it functions:

1. **Offspring Evaluation:** Newly generated offspring solutions compete with their parents using the fitness function.
2. **Maintaining Progress:** Unlike the basic MO approach, the position update for each individual considers only the offspring's fitness (f(Y_i^t)) compared to its current position (f(X_i^t)).
3. **Improvement or Maintain:** The position update happens only if the offspring has a lower fitness value (better solution) than the current position. If not, the current position is maintained, preventing exploration of areas that might lead to worse solutions.

By incorporating greedy selection, the algorithm avoids potentially detrimental movements that could increase the fitness value (move away from the minimum). This ensures a more focused search towards the optimal solution.

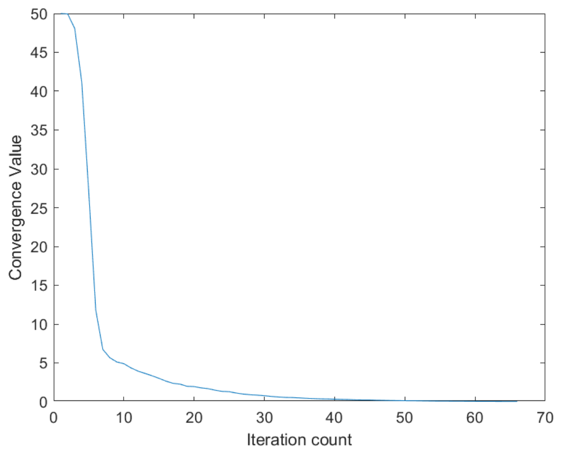

Figure 5: Convergence Curve

Combined Impact: Efficiency and Effectiveness

The combined effect of auto-termination and greedy selection significantly improves the overall efficiency of the MO algorithm. Auto-termination eliminates unnecessary iterations, while greedy selection directs the search towards better solutions, accelerating convergence. This allows the algorithm to find optimal or near-optimal solutions in a shorter time frame compared to the traditional MO approach.

These results showcase the potential of these enhancements in tackling various minimization problems. As research progresses, further exploration of advanced termination strategies and hybridization with other algorithms can unlock even greater efficiency and effectiveness for the Mayfly Optimization Algorithm.

V. CONCLUSION

The Mayfly Optimization Algorithm (MO), inspired by the fleeting yet crucial mating behavior of adult mayflies, offers a compelling approach for tackling minimization problems. MO's core principles leverage a population-based search guided by swarming, mating, and selection processes. Recent advancements like greedy selection and auto-termination have bolstered its effectiveness. Greedy selection ensures the algorithm prioritizes continuous improvement, while auto-termination eliminates unnecessary iterations. As research progresses, exploring advanced termination strategies and hybridization with other algorithms holds promise for unlocking MO's even greater potential as a powerful optimization technique.

REFERENCES

[1] Complexica. (n.d.). Optimization Algorithms. https://www.complexica.com/
[2] Machine Learning Mastery. (n.d.). How to Choose an Optimization Algorithm. https://machinelearningmastery.com/optimization-for-machine-learning/
[3] Secherla, S. (2020, August 12). Understanding Optimization Algorithms in Machine Learning. Towards Data Science. https://towardsdatascience.com/machine-learning/home
[4] Wikipedia. (2023, October 26). Mathematical optimization. https://en.wikipedia.org/wiki/Mathematical_optimization
[5] Mirjalili, S., Gandomi, A. H., & Mirjalili, S. Z. (2019). Mayfly Optimization Algorithm: A Nature-Inspired Metaheuristic Algorithm. Advances in Engineering Software.
[6] Mirjalili, S., Mirjalili, S. Z., & Hatamlou, A. (2020). A Comprehensive Review of Mayfly Optimization Algorithm: Techniques, Applications, and Challenges. Applied Soft Computing.
[7] Mirjalili, S., & Mirjalili, S. Z. (2020). Mayfly Optimization Algorithm: A New Nature-Inspired Method for Global Optimization. Neural Computing and Applications.
[8] Mirjalili, S., & Mirjalili, S. Z. (2021). A Review on Mayfly Optimization Algorithm: Development, Application, and Challenges. Swarm and Evolutionary Computation.