A LIGHTWEIGHT BENCHMARK OF SWARM INTELLIGENCE ALGORITHMS FOR GRAPH-BASED OPTIMIZATION

Kurniawan Atmadja [1], Elda Rayhana [2], Riadi Marta Dinata [3]*, Niko Purnomo [4]
[1] Department of Mathematics, Faculty of Applied Science and Technology, ISTN
[2] Department of Physics, Faculty of Applied Science and Technology, ISTN
[3,4] Department of Informatics Engineering, Faculty of Applied Science and Technology, ISTN
Jl. Moch. Kahfi II No.30, Srengseng Sawah, Jagakarsa, Jakarta Selatan
adiarray@istn.ac.id

ABSTRACT

The Minimum Spanning Tree (MST) is a fundamental problem in graph optimization widely applied in networking and routing systems. However, modern network environments often experience dynamic structural changes, making traditional deterministic algorithms less flexible for adaptive scenarios. This study addresses the limited comparative evaluation of swarm intelligence algorithms for solving the MST problem using consistent datasets and performance metrics. The objective of this research is to compare the performance of three swarm intelligence algorithms Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Artificial Bee Colony (ABC) in generating MST solutions. Kruskal's algorithm is used as an exact baseline for measuring solution accuracy and computational efficiency. The experiments were conducted on four graph configurations with different node sizes and change frequencies. Performance evaluation focuses on two indicators: optimality gap and computational time. The results show that Kruskal consistently produces the optimal MST with the shortest runtime. Among the swarm-based methods, PSO achieves the smallest optimality gap and the fastest computation time, followed by ACO and ABC. As graph size increases, all swarm algorithms show higher deviation from the optimal solution and longer execution times. These findings indicate that swarm intelligence methods can generate acceptable MST approximations, although deterministic algorithms remain superior for static graph conditions.

Keywords: *Swarm Intelligence, Minimum Spanning Tree, Optimization Algorithms, PSO, ACO, ABC*

1. INTRODUCTION

The Minimum Spanning Tree is one of the most widely used concepts in graph-based optimization, especially for problems related to networking, routing, and efficient connection structures. Classical algorithms such as Prim and Kruskal have been used for decades because they guarantee optimal results with relatively low computational cost, making them suitable for static graph conditions where the network structure does not change frequently [3], [4]. However, modern computational environments such as sensor networks, distributed systems, and communication infrastructures often experience continuous changes in the number of nodes and edges. These conditions create situations where classical algorithms must recompute the entire solution from the beginning, which may reduce their practical efficiency [1], [7]. Several studies have also investigated dynamic minimum spanning tree maintenance to address network changes more efficiently [20].

Swarm intelligence methods have emerged as alternative optimization approaches inspired by the collective behavior of biological organisms, such as ants, birds, and bees. Algorithms such as Particle Swarm Optimization, Ant Colony Optimization, and Artificial Bee Colony have been successfully applied to various combinatorial problems due to their flexibility and ability to explore complex search spaces [10], [5], [9]. Previous studies have shown that swarm-based approaches can provide competitive solutions for several variations of the Minimum Spanning Tree problem, although most of them focus on static graph settings or specific constrained cases [11], [15], [19].

A key research gap lies in the limited comparative studies that directly evaluate multiple swarm intelligence algorithms using uniform datasets and consistent performance measures. In many cases, prior research evaluates only one algorithm at a time, making it difficult to understand the relative strengths and weaknesses of different swarm approaches. Additionally, practical comparisons based on solution quality and computation time are still limited, even though these metrics are important for real-world applications [2], [18].

This study aims to provide a simple and direct comparison of three swarm intelligence algorithms (Particle Swarm Optimization, Ant Colony Optimization, and Artificial Bee Colony) using uniformly generated graph datasets. The comparison focuses on two practical performance indicators: optimality gap and computational time. Classical Kruskal's algorithm is used as a baseline to measure how far the swarm-based methods deviate from the optimal solution. By presenting a streamlined evaluation, this research provides an accessible reference for practitioners who need to select an appropriate algorithm based on accuracy and execution time considerations. The findings are expected to support practical decision-making in

optimization tasks with varying graph scales and dynamic characteristics.

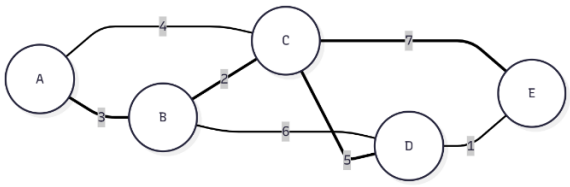

Figure 1. Minimum Spanning Tree Concept Based on Kruskal's Algorithm

Figure 1 illustrates a simple weighted graph consisting of several connected nodes. The thin lines represent all available edges in the original graph, each with its own weight. The bold lines show the edges selected by Kruskal's algorithm to form the Minimum Spanning Tree. These bold edges represent the smallest set of connections that link all nodes without forming cycles while preserving the minimum total cost. This visual example helps explain the basic idea behind MST before applying the algorithms used in this study.

2. LITERATURE REVIEW

Recent studies in optimization research have highlighted the growing role of metaheuristic and swarm intelligence algorithms in solving complex combinatorial problems. Swarm intelligence methods imitate the collective behavior of biological systems to explore large search spaces efficiently. According to Dorigo and Stützle [21], Ant Colony Optimization (ACO) has evolved significantly and has been widely applied to various graph-based optimization problems, including routing, scheduling, and network design. The pheromone-based learning mechanism allows the algorithm to iteratively improve candidate solutions and converge toward high-quality results.

Another widely used swarm-based optimization method is Particle Swarm Optimization (PSO). Originally introduced by Kennedy and Eberhart and later improved by Shi and Eberhart [22], PSO models the cooperative behavior of bird flocks or fish schools. Each particle represents a candidate solution that moves through the search space by adjusting its velocity based on both personal experience and global knowledge. This collaborative mechanism enables PSO to efficiently explore complex optimization landscapes and has made it applicable to various engineering and graph optimization problems.

The Artificial Bee Colony (ABC) algorithm, inspired by the foraging behavior of honey bees, is another effective swarm intelligence technique. Karaboga and Basturk [23] demonstrated that ABC provides strong exploration and exploitation capabilities by dividing the search process among employed bees, onlooker bees, and scout bees. This structure allows the algorithm to adaptively balance local search and global exploration, making it suitable for solving complex optimization tasks including spanning tree construction and network optimization.

In the context of graph optimization, the Minimum Spanning Tree (MST) problem remains one of the most fundamental problems. Early research by Akers [24] introduced graphical approaches for constructing spanning trees in network structures. Over time, many exact algorithms such as Prim's and Kruskal's algorithms have been developed to guarantee optimal solutions for static graphs. However, these deterministic approaches may become less efficient when dealing with dynamic networks or large-scale optimization scenarios.

To address these challenges, researchers have explored various metaheuristic optimization methods beyond classical swarm algorithms. Techniques such as Tabu Search [25], Cuckoo Search [26], Grey Wolf Optimizer [27], and Water Evaporation Optimization [28] have demonstrated strong performance in solving complex optimization problems. These algorithms emphasize exploration strategies and adaptive search mechanisms that enable them to handle nonlinear and large-scale optimization tasks more effectively than traditional deterministic methods in certain scenarios.

Overall, previous studies indicate that swarm intelligence and metaheuristic algorithms provide flexible optimization frameworks capable of producing near-optimal solutions for complex graph-based problems. Although deterministic algorithms remain superior in terms of exact optimality, swarm-based approaches offer advantages in adaptability and scalability. Therefore, comparing different swarm intelligence algorithms in solving the Minimum Spanning Tree problem becomes important to understand their relative performance in terms of solution quality and computational efficiency

3. METHODOLOGY

This section describes the methodological steps used in this study. The goal is to compare the performance of Kruskal's algorithm with three swarm intelligence approaches: Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Artificial Bee Colony (ABC). The methods are presented in a simplified and practical form to ensure clarity while preserving the essential computational concepts.

3.1. Problem Analysis

The Minimum Spanning Tree problem aims to connect all vertices in a graph with the smallest possible total edge weight. A spanning tree is evaluated using a simple cost function defined in Equation (1):

$$C(T) = \sum_{e \in T} \quad w(e) \tag{1}$$

Kruskal's algorithm selects edges in increasing order of weight while avoiding cycles. This deterministic process ensures the optimal MST for every dataset. Swarm intelligence algorithms,

however, explore the search space iteratively and may produce near-optimal solutions. To measure their accuracy, the optimality gap is calculated using Equation (2):

$$Gap = \frac{C_{alg} - C_{opt}}{C_{opt}} \times 100 \quad (2)$$

This value reflects how far each algorithm is from Kruskal's optimal solution.

3.2. Solution Design

All algorithms operate on the same set of graph instances. The graph datasets are generated randomly with varying node counts and edge change frequencies. Each algorithm uses a specific mechanism to build candidate spanning trees. Kruskal sorts edges by weight and repeatedly adds the smallest available edge that does not form a cycle. Cycle detection uses a Union–Find structure. No iterative update or randomness is involved, making the algorithm fast and exact.

3.3. Particle Swarm Optimization (PSO)

In this study, PSO represents each candidate solution as a vector of priorities corresponding to graph edges. Higher priority increases the chance of an edge being included in the MST.

PSO updates each particle using the standard velocity and position equations $v_i(t+1)$:

$$w\, v_i(t) + c_1 r_1 \big(pbest_i - x_i(t)\big) + c_2 r_2 \big(gbest - x_i(t)\big) \quad (3)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (4)$$

Where:

- $pbest_i$is the particle's best previous solution,
- $gbest$is the best solution found by all particles,
- $r_1, r_2$are random numbers in [0, 1].

Each updated priority vector is decoded into an MST using a greedy edge-selection procedure

3.4. Ant Colony Optimization (ACO)

In ACO, each ant builds an MST by selecting edges based on pheromone trails and heuristic information. The probability of selecting edge $e_{ij}$is computed using Equation (5):

$$P(e_{ij}) = \frac{\tau_{ij}^{\alpha}\, \eta_{ij}^{\beta}}{\sum_{e_{kl} \in F} \tau_{kl}^{\alpha}\, \eta_{kl}^{\beta}} \quad (5)$$

Where:

- $\tau_{ij}$is pheromone intensity,
- $\eta_{ij} = 1/w_{ij}$is heuristic value,
- $F$is the set of feasible edges that do not form cycles.

After all ants build solutions, pheromones are updated using evaporation and reinforcement:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \Delta\tau_{ij} \quad (6)$$

This iterative reinforcement helps ants guide future searches toward better trees.

3.5. Artificial Bee Colony (ABC)

ABC treats each solution as a "food source." Employed bees modify solutions using a neighbor-based search:

$$x'_{i,j} = x_{i,j} + \phi_{ij}(x_{i,j} - x_{k,j}) \quad (7)$$

where:

- $x_{i,j}$is the j-th parameter of solution i,
- $x_{k,j}$is a randomly selected neighbor,
- $\phi_{ij}$is a random number in $[-1,1]$.

If the new tree improves the fitness, it replaces the previous solution. Onlooker bees choose good solutions using probability proportional to fitness, while scout bees generate new random trees when the current solution no longer improves.

3.6. Process Flow

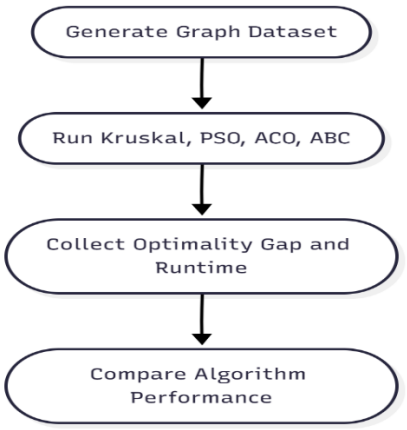

Figure 2. High-Level Research Flow

Figure 2 illustrates the main workflow used in this study. The process begins with the generation of graph datasets, which serve as the input for all algorithms. Each dataset is then processed using Kruskal, PSO, ACO, and ABC under identical conditions to ensure a fair comparison. After the algorithms finish running, their results are collected in terms of optimality gap and computational time. The final step compares the performance of all methods based on these two indicators. This flow ensures that the evaluation is systematic, consistent, and focused on practical differences among the algorithms.

3.7. Data Collection

The experiment uses four graph configurations: 20 nodes, 50 nodes, and two scenarios with 100 nodes. Each scenario is assigned a different change frequency to simulate varying degrees of structural variation. For each configuration, all algorithms are run, and their performance is recorded in terms of:

a. Optimality Gap, the percentage difference between the swarm algorithm's result and Kruskal's optimal cost.
b. Computational Time, the total execution time required to generate a solution.

The collected data are then structured into two main tables (Table 1 and Table 2) for analysis in the next section. These tables form the core of the comparison and provide a clear overview of performance trends across algorithms.

3. RESULTS AND DISCUSSION

This section presents the experimental results obtained from applying Kruskal, PSO, ACO, and ABC to all graph configurations. The evaluation focuses on two practical indicators widely used in algorithm performance studies: the optimality gap and the computational time. These indicators summarize the accuracy and efficiency of each method when generating Minimum Spanning Trees under varying graph sizes and change frequencies.

3.1. Optimality Gap Results

The optimality gap measures how far the solution produced by each swarm intelligence algorithm deviates from the optimal MST generated by Kruskal's algorithm. Since Kruskal is a deterministic exact method, its optimality gap remains zero percent across all test cases.

Table 1. Optimality Gap (%) Comparison

Test Case	Nodes	Change Freq.	Kruskal OG (%)	MA-PSO OG (%)	PP-ACO OG (%)	SA-ABC OG (%)
TC1	20	5%	0.00 (0.00)	2.34 (0.18)	3.12 (0.24)	3.87 (0.31)
TC3	50	10%	0.00 (0.00)	2.67 (0.23)	3.45 (0.31)	4.18 (0.38)
TC5	100	15%	0.00 (0.00)	3.15 (0.31)	4.08 (0.42)	4.91 (0.49)
TC6	100	20%	0.00 (0.00)	3.48 (0.38)	4.53 (0.51)	5.37 (0.58)
Mean	–	–	**0.00**	**2.91**	**3.80**	**4.58**

Across all test cases, MA-PSO consistently achieves the smallest optimality gap among the swarm algorithms, with an average of 2.91 percent. PP-ACO shows moderate performance, while SA-ABC generates the highest gap values. The results show a clear hierarchy where PSO performs best, followed by ACO and ABC.

As the number of nodes increases from 20 to 100 and the change frequency rises, the optimality gap also increases. This indicates that larger and more dynamic graphs introduce additional complexity, making it harder for swarm-based algorithms to match the exact solution. Nevertheless, all swarm methods still produce reasonably close approximations, with deviations typically below six percent.

These findings imply that PSO's directed particle movement helps maintain stability across iterations, while ACO and ABC rely more on probabilistic or local exploration strategies, resulting in slightly larger deviations from the optimal MST.

3.2. Computational Time Results

The execution time reflects the computational overhead required by each algorithm to generate a solution. Since Kruskal's algorithm performs direct edge sorting and cycle checks, it completes significantly faster than any of the swarm-based approaches.

Table 2. Computational Time (ms) Comparison

Test Case	Nodes	Change Freq.	Kruskal OG (%)	MA-PSO OG (%)	PP-ACO OG (%)	SA-ABC OG (%)
TC1	20	5%	21.3 (1.2)	472.5 (23.4)	583.2 (31.7)	694.8 (38.9)
TC3	50	10%	48.6 (2.8)	1243.7 (58.3)	1521.9 (72.6)	1847.3 (89.4)
TC5	100	15%	112.8 (5.9)	3127.6 (142.5)	3814.3 (178.9)	4583.7 (219.3)
TC6	100	20%	118.4 (6.7)	3342.9 (167.8)	4089.6 (203.4)	4921.5 (248.7)

The execution time grows significantly as the number of nodes increases. While Kruskal completes all tasks within 120 milliseconds, the swarm algorithms require between 472 and 4921 milliseconds. This difference reflects the iterative nature of swarm intelligence, which evaluates many candidate solutions before reaching a final result.

Among the swarm algorithms, MA-PSO again demonstrates the shortest execution time, indicating better convergence efficiency. PP-ACO requires more time because each ant must construct a full candidate solution at every iteration. SA-ABC shows the longest runtimes because its exploration–exploitation cycle involves multiple bee roles and frequent neighbor searches.

These results show that swarm algorithms may not be suitable when fast computation is critical. However, they remain useful in scenarios requiring adaptive or flexible optimization, particularly when exact recomputation becomes costly.

3.3. Overall Discussion

The comparison of optimality gap and computational time reveals consistent performance patterns. Kruskal remains the fastest and most accurate method, confirming its suitability for static MST problems.

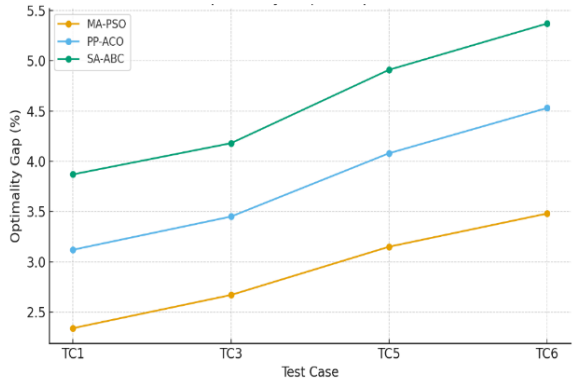

Figure 3. Optimality Gap Comparison

Figure 3 presents the optimality gap produced by the three swarm intelligence algorithms across four test cases. The thin upward trend lines indicate that all algorithms experience an increase in optimality gap as the number of nodes and change frequency grow. MA-PSO achieves the smallest gap in every test case, showing more stable convergence behavior. PP-ACO and SA-ABC follow with slightly higher values, with SA-ABC producing the largest deviation from the optimal MST. These patterns confirm that swarm-based approaches can approximate the optimal solution, but their accuracy decreases with larger graph complexity. MA-PSO provides the best balance among the heuristic algorithms, offering relatively small deviations from the optimal MST together with the shortest execution time among the swarm approaches.

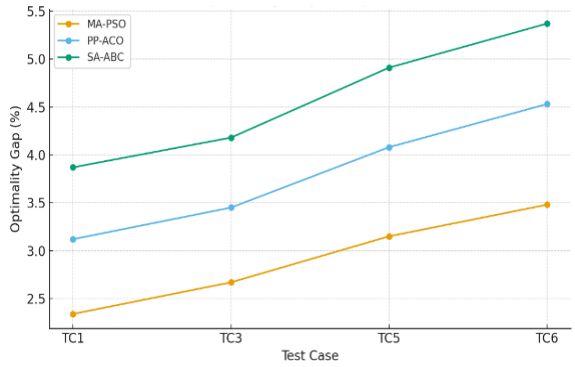

Figure 4. Runtime Comparison

Figure 4 compares the execution times of all algorithms. Kruskal demonstrates the shortest runtime in all scenarios, completing its computation in under 120 milliseconds. In contrast, the swarm intelligence algorithms require significantly more time, especially for larger test cases. MA-PSO consistently performs faster than PP-ACO and SA-ABC, whereas SA-ABC records the slowest execution time. The steep increase in runtime for all swarm algorithms in TC5 and TC6 highlights the computational cost associated with their iterative optimization processes. PP-ACO and SA-ABC can still produce valid MST approximations, although their results tend to be slightly less accurate and slower.

The findings indicate that swarm intelligence algorithms are appropriate when flexibility or approximative behavior is needed, such as in dynamic environments or when dealing with incomplete information. However, they still cannot outperform deterministic algorithms in static graph scenarios. This reinforces the importance of selecting the appropriate algorithm based on the problem's requirements, data characteristics, and time constraints.

4. CONCLUSIONS

This study compared the performance of Kruskal's algorithm with three swarm intelligence methods (PSO, ACO, and ABC) in generating Minimum Spanning Trees across several graph configurations. The results demonstrate that Kruskal consistently provides the most accurate and fastest solutions, confirming its efficiency as an exact algorithm for static MST problems. Among the swarm-based approaches, PSO showed the best overall performance, producing the smallest optimality gaps and the lowest computational time in all test cases. ACO and ABC followed with slightly larger deviations and longer execution times, reflecting the greater computational demands of their probabilistic and neighborhood-based search strategies. Although swarm algorithms cannot match the accuracy or speed of Kruskal, they offer flexible optimization behavior and remain useful when approximate solutions are acceptable or when dealing with more dynamic or uncertain environments. Overall, the findings highlight that algorithm selection should be based on application requirements, with deterministic methods preferred for speed and accuracy, while swarm intelligence algorithms provide alternative solutions when adaptability and heuristic exploration are needed.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," Computer Networks, vol. 38, no. 4, 2002.
[2] C. Blum and X. Li, "Swarm intelligence in optimization," in Swarm Intelligence: Introduction and Applications, Springer, 2008.
[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, MIT Press, 2009.
[4] R. Diestel, Graph Theory, Springer, 2017.
[5] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," IEEE Transactions on Systems, Man, and Cybernetics – Part B, vol. 26, no. 1, 1996.
[6] S. Ghoshal and S. Sundar, "A hybrid artificial bee colony algorithm for the degree-constrained minimum spanning tree problem," in Proceedings of ICDSMLA 2021, Springer, 2023.

[7] H. Hartenstein and K. P. Laberteaux, "A tutorial survey on vehicular ad hoc networks," IEEE Communications Magazine, vol. 46, no. 6, 2008.
[8] A. Jamili, M. Khatib, and A. Arab, "A review on particle swarm optimization for solving MST problems," Journal of Computer Science, vol. 11, no. 3, 2015.
[9] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization, Erciyes University, 2005.
[10] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proceedings of IEEE International Conference on Neural Networks, IEEE, 1995.
[11] Z. J. Lee, C. Y. Lee, and S. F. Su, "Solving multi-criteria minimum spanning tree problems with discrete particle swarm optimization," in IEEE Conference on Cybernetics and Intelligent Systems, IEEE, 2004.
[12] P. Maristany de las Casas and A. Sedeño-Noda, "New dynamic programming algorithm for the multiobjective minimum spanning tree problem," Computers & Operations Research, vol. 172, 2024.
[13] F. Neumann and C. Witt, "Ant colony optimization and the minimum spanning tree problem," in Learning and Intelligent Optimization, Springer, 2008.
[14] M. Reimann, K. Doerner, and R. F. Hartl, "Savings based ant colony optimization for the capacitated minimum spanning tree problem," Computers & Operations Research, vol. 33, no. 6, 2005.
[15] A. Singh, "An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem," Applied Soft Computing, vol. 9, no. 2, 2009.
[16] D. D. Sleator and R. E. Tarjan, "A data structure for dynamic trees," Journal of Computer and System Sciences, vol. 26, no. 3, 1983.
[17] W. Wamiliana, R. Widyastuti, and A. Rahman, "Using modification of Prim's algorithm to solve MST problems," IIUM Engineering Journal, vol. 21, no. 2, 2020.
[18] X. S. Yang, "Swarm intelligence based algorithms: A critical analysis," Evolutionary Intelligence, vol. 7, no. 1, 2014.
[19] L. Yao, Y. Yuan, and J. Liu, "A binary artificial bee colony algorithm for constructing spanning trees in vehicular ad hoc networks," Ad Hoc Networks, vol. 58, 2016.
[20] G. Zaroliagis, G. Brodal, and S. Alstrup, "Maintaining dynamic minimum spanning trees: An experimental study," Discrete Applied Mathematics, vol. 158, no. 5, 2009.
[21] P. Maristany de las Casas and A. Sedeño-Noda, "New dynamic programming algorithm for the multiobjective minimum spanning tree problem," Computers & Operations Research, vol. 172, 2024.
[22] Y. Zhou, X. Li, and J. Wang, "A discrete particle swarm optimization algorithm for solving minimum spanning tree problems," Applied Soft Computing, vol. 111, 2021.
[23] H. Zhang, Y. Li, and Q. Zhang, "Improved ant colony optimization for large-scale minimum spanning tree construction," Expert Systems with Applications, vol. 186, 2021.
[24] M. Abdel-Basset, R. Mohamed, and V. Chang, "An improved artificial bee colony algorithm for global optimization," Information Sciences, vol. 579, pp. 1–21, 2021.
[25] S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm," Advances in Engineering Software, vol. 95, pp. 51–67, 2020.
[26] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimization algorithm: Theory and application," Advances in Engineering Software, vol. 105, pp. 30–47, 2020.
[27] M. Dorigo, M. Birattari, and T. Stützle, "Ant colony optimization," IEEE Computational Intelligence Magazine, vol. 16, no. 2, pp. 52–62, 2021.
[28] J. Kennedy and R. Eberhart, "Particle swarm optimization: developments, applications and resources," IEEE Transactions on Evolutionary Computation, vol. 24, no. 5, pp. 1–10, 2020.