Civil and Project Journal, 2024, 6(11), 78-87
https://doi.org/10.22034/cpj.2024.482516.1326

Civil and Project Journal
http://www.cpjournals.com/

Research Article

Analyzing the impact of ant colony optimization parameters for path searching behavior

Soheil Rezashoar[1]*, Amir Abbas Rassafi[2]

1*- PhD Student, Department of Transportation Planning, Faculty of Engineering, Imam Khomeini International University, Qazvin, Iran.

2- Professor, Department of Transportation Planning, Faculty of Engineering, Imam Khomeini International University, Qazvin, Iran.

Received: 08 September 2024; Revised: 27 September 2024; Accepted: 14 October 2024; Published: 14 October 2024

Abstract

Ant-inspired metaheuristic algorithms, such as Ant Colony Optimization (ACO), are dependable for addressing intricate problems in discrete and continuous domains. This study examines the influence of the pheromone significance factor (α), heuristic importance factor (β), and pheromone decay rate (ρ) on the effectiveness of ACO for path-searching. We analyze the algorithm's convergence rate and effectiveness in identifying the shortest path by simulating various parameter configurations on a standard graph. The value α= 2 was chosen based on prior research on the behavior of real ants. Our simulations demonstrated that α= 2 is a superior choice to α= 1, which the naïve approach would recommend. The experiments demonstrated that setting β to 1 and ρ to 10% resulted in the optimal convergence speed and the minor average path lengths. Also, by examining the effect of the number of ants on the convergence of the simulation, it was found that the selection of more ants shows more paths. Using more ants for the initial stop leads to a marginal decrease in the average path length.

Keywords: *Ant Colony Optimization, ACO, Path-Searching, Metaheuristics, Parameter Tuning, Discrete Optimization, Convergence Analysis.*

Cite this article as: Rezashoar,S. and Rassafi,A. A. (2025). Analyzing the impact of ant colony optimization parameters for path searching behavior. (e207799). *Civil and Project, 6*(11), e207799 . https://doi.org/10.22034/cpj.2023.04.03.1133

ISSN: 2676-511X /

Corresponding author E-mail address: soheilshoar@edu.ikiu.ac.ir*

1. Introduction

Nature has long been a source of inspiration for problem-solving, providing sophisticated solutions to problems by showcasing adaptation, cooperation, and self-organization observed in biological, physical, and social systems (Bäck, 2018). Nature provides simplistic approaches to solve many complex and challenging problems encountered in real life. Algorithms that draw inspiration from natural systems and processes are called "Nature Inspired Algorithms" (NIAs) (Agarwal & Mehta, 2014; Vassiliadis & Dounias, 2009). These algorithms facilitate the attainment of answers by readily adjusting to natural dynamic fluctuations. By drawing inspiration from biological or natural systems, complex optimization problems can be solved (Ostfeld, 2011). More than 150 bio-inspired algorithms have been written about so far (Bonyadi & Michalewicz, 2017). These include Genetic Algorithms (GA) (Forrest, 1996), Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 1995; Shami et al., 2022), Ant Colony Optimization (ACO) (Dorigo & Di Caro, 1999), Bee Colony Optimization (BCO) (Teodorovic & Dell'Orco, 2005), and many more. However, each of these algorithms has its factors that affect how well they work. In the past ten years (Nannen & Eiben, 2006), many scholars have been interested in tuning (or setting) the parameters. This is because, even if the algorithm works well, setting the parameters to the wrong values can result in a bad solution. For instance, in (Hassanat et al., 2019), the researchers said that GA search must consider factors like mutation rates, crossover rates, and population. In the same way, the three central control factors of the PSO algorithm—inertia weight, cognitive acceleration coefficient, and social acceleration coefficient—have a direct effect on how well it searches. As explained in (Carlisle; Trelea, 2003; Van den Bergh & Engelbrecht, 2006), setting the PSO control parameters ahead of time may make it work better by making it more sensitive to these parameters. Like other bio-inspired algorithms, the Ant Colony Optimization (ACO) requires an initial parameter configuration before commencement. In this paper, we evaluate the effects of the ACO settings on path-searching behavior problem.

The Ant Colony Optimization (ACO) is a widely used technique in the field of swarm intelligence, which draws inspiration from the foraging behavior of actual ant colonies (Colorni et al., 1991; Dorigo et al., 1999; Dorigo & Gambardella, 1997). The ACO is a method in which a group of artificial ants builds solutions based on artificial pheromone trails and heuristic knowledge. It has been successfully applied to solve a wide range of intricate optimization challenges (Mavrovouniotis, Ellinas, et al., 2019; Mavrovouniotis, Li, et al., 2019; Mavrovouniotis & Yang, 2013). The ACO algorithms aim to synchronize communities of artificial ants by employing a type of artificial stigmergy. These algorithms are specifically designed for discrete optimization issues and have been successfully used to various problem-solving tasks, including the Traveling Salesman Problem (TSP) and Data Network Routing (DNR) (Proctor & Khakoo, 2005), Quadratic Assignment (QA) (Maniezzo & Colorni, 1999), Efficient Graph Search (EGS) (Wagner et al., 1998), and telecommunication network problems (Schoonderwoerd et al., 1997). The ACO algorithm is mostly derived from the foraging behavior of ant colonies. Typically, initially, ants haphazardly explore their surroundings in order to find food and subsequently bring a portion of the food back to their nest. In addition, they deposit a pheromone along the path they have discovered. The longevity of pheromones deposited on their trail is contingent upon the caliber and magnitude of the food supply, which dissipates gradually over time. Persisting pheromones on a pathway can influence other ants to trail the same path. Within a brief duration, the majority of ants are capable of navigating the shorter route indicated by the most potent pheromone. Algorithm 1 provides a comprehensive pseudocode of the ACO algorithm (Rezvanian et al., 2023).

Algorithm 1. Pseudocode of Standard Ant Colony Optimization (ACO) for Solving Optimization Problems

```
Begin
    Initialize user parameters of ACO and generate the initial population of the ant colony
While the stopping condition is not met do
        Position each ant in a starting node
Repeat
For each do
                Choose the next node by applying the state transition rule
                Apply step-by-step pheromone update
End for
 Until every ant has built a solution
        Update the best solution
        Apply offline pheromone update
 End While
End
```

Through the running of the algorithm, ants first produce different solutions randomly. Then, the solutions are improved by updating the pheromones according to the type of the problem and graph traversal, pheromones set on vertices or edges of the graph. Traversing the edge between two nodes i and j depends on the probability of edge, which is calculated as given follows:

$$\boldsymbol{p_{ij}^{k}} = \frac{\boldsymbol{\tau(t)_{ij}^{\alpha}\eta_{ij}^{\beta}}}{\sum_{\boldsymbol{j\epsilon N^{k}(i)}} \boldsymbol{\tau(t)_{ij}^{\alpha}\eta_{ij}^{\beta}}} \quad (1)$$

Where, p_{ij}^{k} is the probability of traversing from the edge between nodes i and j $\tau(t)_{ij}^{\alpha}$ denotes the value of the pheromone trail, η_{ij}^{β} is the heuristic information, and $N^{k}(i)$ is the set of nodes that can be visited by ant k. Before updating the pheromones, it is recommended to perform a local search to improve results. However, the method of updating the pheromones is suggested as given follows (Dorigo & Di Caro, 1999):

$$\boldsymbol{\tau_{ij}(t+1) = (1-\rho).\tau_{ij} + \Delta\tau_{ij}(t)} \quad (2)$$

Where ρ is the evaporation rate of pheromone and $\Delta\tau_{ij}(t)$ denotes the amount of pheromone trail put on the edge between nodes i andj. That amount is calculated according to the quality of the solutions found by the whole colony, which include the edge between nodes i and j.

2. Research Method

2.1 Algorithm Implementation

We utilized Ant Colony Optimization (ACO) in a Python framework. To utilize the interface, one must instantiate an AntColonyOptimization object, providing the necessary parameters through the constructor. Then, the run () function can be invoked to simulate the ant colony and obtain a list containing the lengths of all paths discovered by the ants. Minor adjustments could be implemented to provide precise pathways instead. The essential parameters required to execute the simulation are:

1. A graph represented as an adjacency list.

2. A dictionary containing edge costs.

3. Source and destination nodes within the graph.

Optionally, the user can input the number of ants, the number of iterations, α and β parameters, and a pheromone decay rate ρ. The process of selecting parameters varies depending on the type of optimization problem. Furthermore, parameters may vary depending on the scale of the optimization issue, even if it is of the same type. Nevertheless, the algorithm can execute with the default settings for these parameters. Every individual ant in the colony possesses a memory that is capable of storing information about its path, the total length of the path, and whether it is currently returning to the source or not. The main loop of run () executes for the specified number of iterations. At each step, each ant moves once. The movement of an ant can be either forward or backward. In forward movement, the ant chooses a random node from its neighborhood. By default, it excludes the node it just came from unless it is the only possible node to visit, such as when the ant reaches a dead end in the graph. Loops are allowed during this phase. While moving forward, ants do not deposit pheromones. When an ant reaches its destination, it performs two key actions. First, it reviews its path and removes any loops. Then, it calculates the total length of its path as the sum of the edge costs. Each and then moves the backbone node along its cycle-free path, depositing a pheromone on the traversed edge. The amount of pheromone deposited is the inverse of the ant's total path length. The rationale is that the ant has a single unit of pheromone to deposit on its return path from the destination, so it is distributed uniformly along the path. Upon reaching the source node, the ant reports its path length and is reset, starting another walk in the next iteration. At the end of each iteration, pheromone decay is applied to all edges. All simulations were conducted on the graph shown in Figure 1, extracted from (Parsons, 2005). For simplicity, all edges have unit cost, although the simulation can handle non-uniform costs. Note that the shortest path from source to destination has a length of 5, while the top path has a length of 8. However, the bottom approach has the possibility of more intricate paths, which could include loops. Therefore, even though at the start, ants have a 50% probability of choosing the top or the bottom path, we expect the initial average path length to be closer to 8 than to 5 since not all ants choosing the bottom path will follow the shortest path.

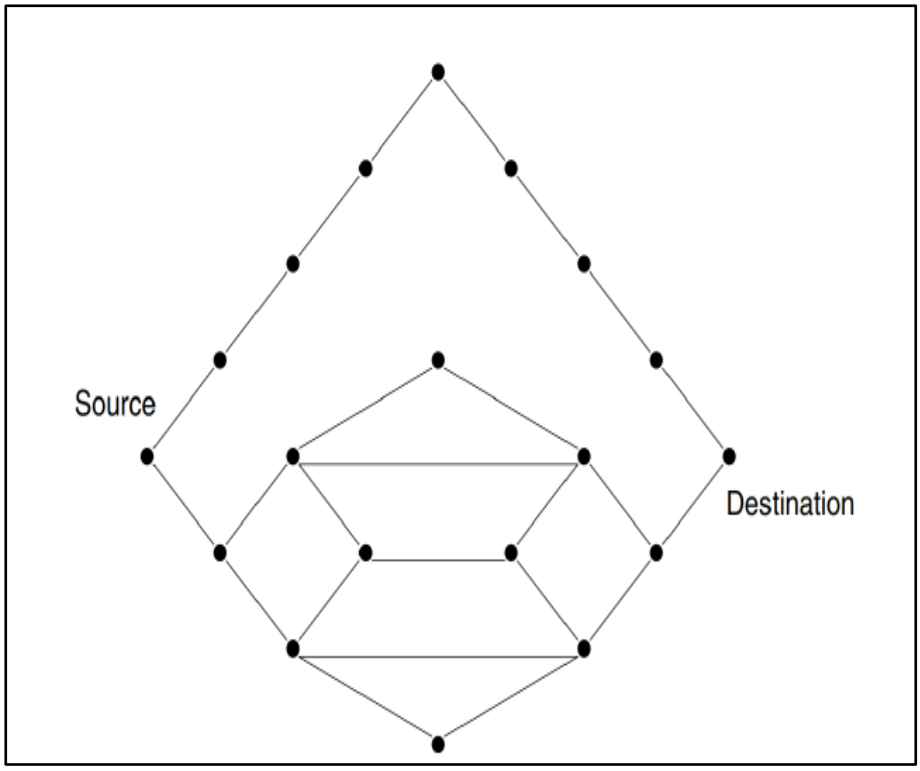

Figure 1. Graph Used in the Experiment (All Edges have Unit Cost.)

2.2 Algorithm Evaluation

12 simulations were carried out, each with different parameter selections. Every simulation comprised 100,000 iterations. The iterations are executed on a Central Processing Unit (CPU) with an Intel i7-9750H, accompanied by a Random Access Memory (RAM) of 16GB, and a Graphics Processing Unit (GPU) of GTX 1660Ti. The overall duration amounted to almost one hour. The customizable parameters of the standard Ant Colony Optimization (ACO) algorithm are listed in Table 1 (Yang et al., 2020).

Table 1. Controllable Parameters in the Ant Colony Optimization (ACO)

Symbol	Parameter	Description
M	**Ant colony size**	**The size of the ant colony plays a crucial role in determining the ideal solution.**
α	**Information heuristic factor**	**The importance of pheromone.**
β	**Expect heuristic factor**	**The degree to which shorter paths within a small area are considered important.**
ρ	**Pheromone volatility factor**	**Pheromone volatility on the path.**

In general, as the size of the ant colony, denoted as M, increases, the algorithm's search capability becomes more robust. However, this also increases the likelihood of generating several duplicate solutions. Excessive iterations of the algorithm near the optimal value will deplete significant resources (Yang et al., 2020). The probability of an ant choosing a local path based on pheromone concentration increases as the value of the information heuristic factor α increases. As the predicted heuristic factor β increases, the likelihood of ants selecting a shorter local path increases, making the algorithm more inclined towards greediness and reducing the unpredictability of the search (Yang et al., 2020). The pheromone volatility factor ρ is limited to values between 0 and 1. When the value of ρ is too little, the pheromone on the path will not evaporate quickly enough, leading to an excessive amount of pheromone on the path. This excessive pheromone will negatively impact the efficiency of the algorithm's convergence. When the value of ρ is excessively high, the pheromone on the path cannot be preserved, resulting in the loss of experiential information from past iterations for the ant colony (He et al., 2017).

3. Results and Discussion

The Ant Colony Optimization (ACO) in swarm intelligence exemplifies the fundamental traits of this paradigm. Establishing parameters appropriate for diverse, intricate situations is facile and exhibits considerable robustness. Consequently, it is extensively utilized in pathfinding. As mentioned, parameter setting means finding suitable parameters for algorithms to optimize their performance. Because it controls their behavior, it dramatically impacts the accuracy of ant colony-based algorithms. The ACO has several parameters that control different aspects. This section analyzes the most critical parameters and their impact on the algorithm's performance.

Initially, the study focused on assessing the influence of α value to determine whether α= 1 (representing a simple probability proportionate to the amount of pheromone) or α= 2 (based on actual and behavioral factors (Deneubourg et al., 1990)) was the most optimal choice. The remaining parameters were held constant at β= 1, ρ= 1%, and M= 128. Furthermore, the significance of the heuristic was assessed by examining its exponent β. We conducted tests with β values of 0 (no heuristic used), 1, and 2 for the same rationale as α. The conventional parameters used for these tests were α= 1, ρ= 1%, and M= 128. Furthermore, the pheromone decay rate was manipulated with three different values: ρ= 0 (indicating no pheromone decay), ρ= 1% (representing low evaporation), and ρ= 10% (indicating high evaporation). The values chosen for the parameters of this suite were α= 1, β= 1, and M= 128. Four simulations were conducted, in which the number of ants M was adjusted, doubling between 64 and 512. The initial values for the parameters were α= 1, β= 1, and ρ= 1%.

According to Figure 2, for both values of α, the rolling average of path lengths found by the ants started at about 7, which confirmed our hypothesis that the initial path length would be closer to 8 than to 5. We also note that both averages converge to 5, which is indeed the shortest path length possible on the test graph. However, it is clear that α= 2 converged way quickly. Note the logarithm scale on the horizontal axis: it converged with about 10 times fewer iterations than α= 1. Testing the simulation for different values of β, however, did not significantly alter the results, as shown in Figure 3. If the simulation stops before convergence, it appears that β= 1 gives the smallest average path length, but all three values tested converged at similar points, with around 60,000 paths found. This run had 128 ants, so it averaged about 470 trips per ant for convergence.

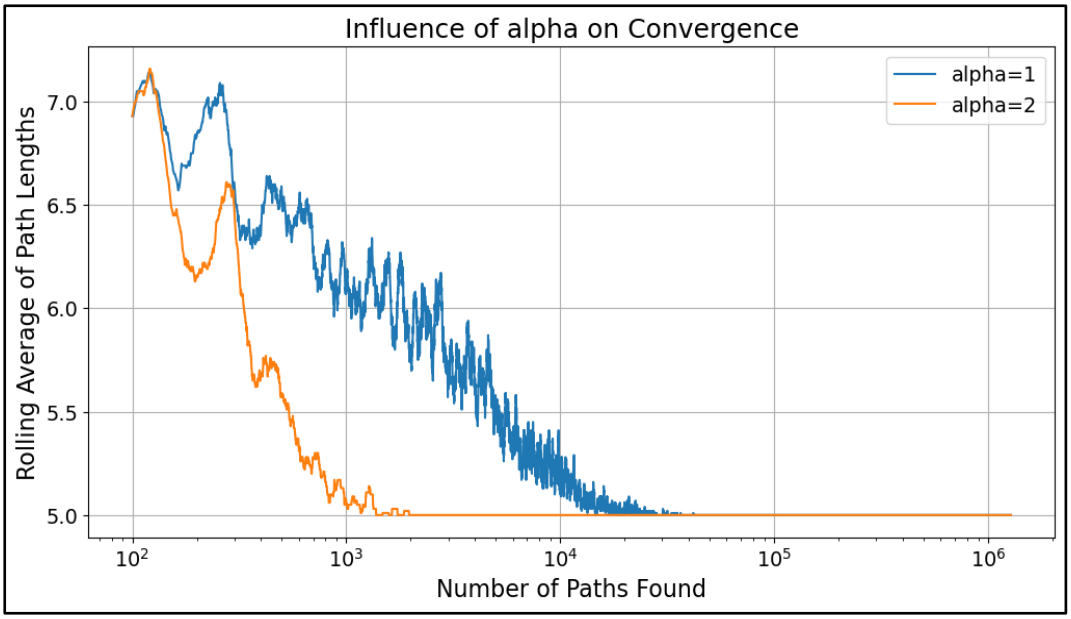

Figure 2. Influence of α in Rolling Average of Path Length

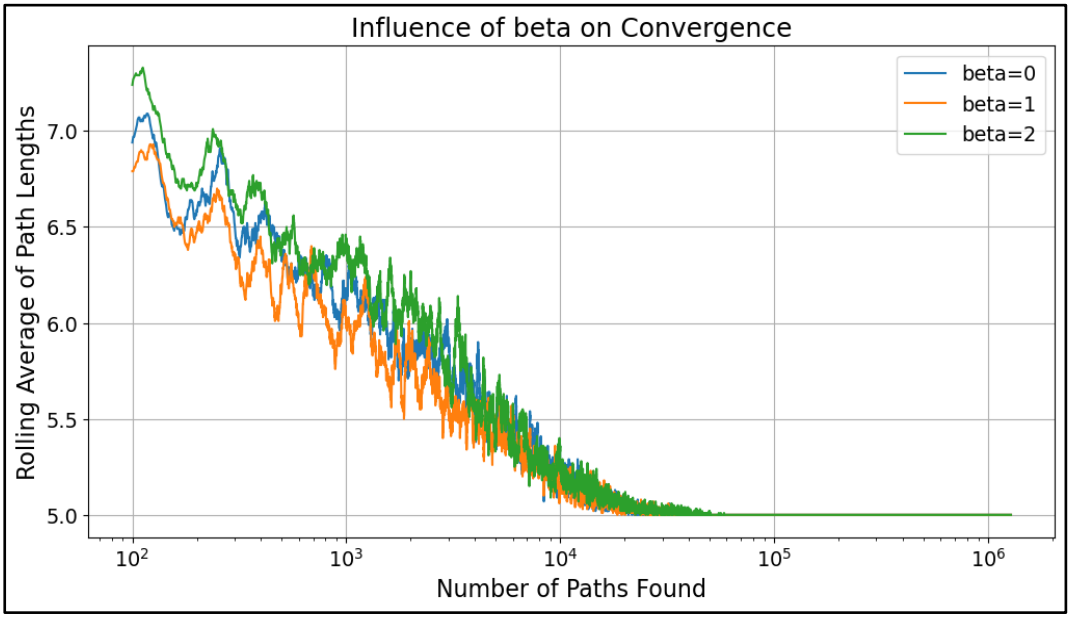

Figure 3. Influence of β in Rolling Average of Path Length

Changing the pheromone decay rate ρ had substantial impacts, as seen in Figure 4. Without pheromone evaporation (ρ= 0), the simulation did not converge even after 100,000 iterations and more than one million paths traveled. The average path length oscillated between 5 and 5.5. It can also be seen that high evaporation rates, around 10% per iteration, yielded better results in the form of faster convergence at about 5,000 paths found. In contrast, low evaporation rates only reached convergence after 60,000 paths. Finally, we can see the effect of the number of ants on the simulation convergence, as shown in Figure 5. For this plot, we scaled the x-axis based on the number of ants since. Naturally, more ants would yield more paths. Remarkably, the algorithm's convergence remains consistent across this range of ant population sizes. However, employing more ants for early stopping does lead to a marginal decrease in average path length.

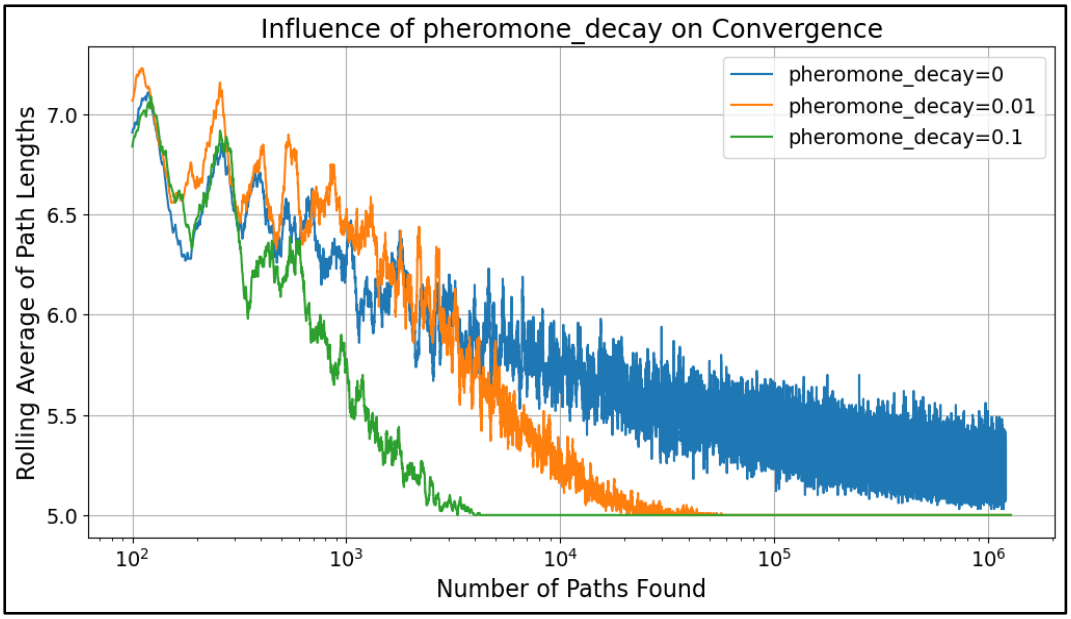

Figure 4. Influence of Pheromone Decay Rate ρ in Rolling Average of Path Length

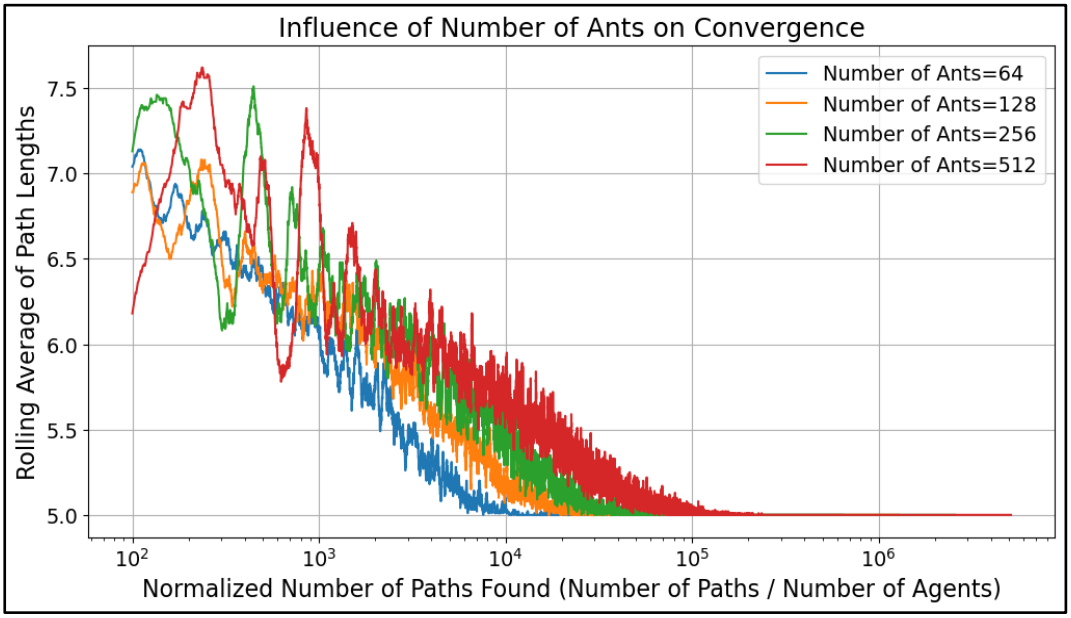

Figure 5. Influence of Number of Ants M in Rolling Average of Path Length

4. Conclusions

Swarm intelligence is a contemporary method for tackling optimization problems. It usually mimics the social behavior observed in birds and animals. Ant Colony Optimization (ACO) is the most widely used type of swarm intelligence because it simulates the search and movement patterns of ants. It shows that one ant in an ant colony is not intelligent, but when they work together to make a whole system, they become smart and can find the best way to get around in a complicated world. Because of this, it is studied and used a lot in path planning. The optimal performance of ACO primarily relies on appropriate settings; hence, many research works study the effect of ant colony algorithm parameters. This study examined the influence of the pheromone significance factor α, heuristic importance factor β, and pheromone decay rate ρ on the effectiveness of an ant colony optimization algorithm for path-finding. The value α= 2 was selected based on research in real ants' behavior (Deneubourg et al., 1990), and our simulations showed it is indeed a better choice than α= 1, the naïve approach would suggest. Experiments showed that β= 1 and ρ= 10% provided the best convergence speed and shortest average path lengths. Also, by examining the effect of the number of ants on the simulation's convergence, it was found that selecting more ants shows more paths. Using more ants for the initial stop leads to a marginal decrease in the average path length. Future research should involve testing additional graph types, including those with non-uniform edge costs or increased nodes.

References

Agarwal, P., & Mehta, S. (2014). Nature-inspired algorithms: state-of-art, problems and prospects. International Journal of Computer Applications, 100(14), 14-21.

Bäck, T. (2018). Evolutionary computation 1: Basic algorithms and operators. CRC press.

Bonyadi, M. R., & Michalewicz, Z. (2017). Particle swarm optimization for single objective continuous space problems: a review. Evolutionary computation, 25(1), 1-54.

Carlisle, A. G., 2001. An off-the-shelf PSO. Proceedings of the Workshop on Particle Swarm Optimization.

Colorni, A., Dorigo, M., & Maniezzo, V. (1991). Distributed optimization by ant colonies. Proceedings of the first European conference on artificial life.

Deneubourg, J.-L., Aron, S., Goss, S., & Pasteels, J. M. (1990). The self-organizing exploratory pattern of the argentine ant. Journal of insect behavior, 3, 159-168.

Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406).

Dorigo, M., Di Caro, G., & Gambardella, L. M. (1999). Ant algorithms for discrete optimization. Artificial life, 5(2), 137-172.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on evolutionary computation, 1(1), 53-66.

Forrest, S. (1996). Genetic algorithms. ACM computing surveys (CSUR), 28(1), 77-80.

Hassanat, A., Almohammadi, K., Alkafaween, E. a., Abunawas, E., Hammouri, A., & Prasath, V. S. (2019). Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach. Information, 10(12), 390.

He, J., Sun, X., Li, W., & Chen, J. (2017). A new pheromone update strategy for ant colony optimization. Journal of Intelligent & Fuzzy Systems, 32(5), 3355-3364.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. Proceedings of ICNN'95-international conference on neural networks.

Maniezzo, V., & Colorni, A. (1999). The ant system applied to the quadratic assignment problem. IEEE Transactions on knowledge and data engineering, 11(5), 769-778.

Mavrovouniotis, M., Ellinas, G., & Polycarpou, M. (2019). Electric vehicle charging scheduling using ant colony system. 2019 IEEE Congress on Evolutionary Computation (CEC).

Mavrovouniotis, M., Li, C., Ellinas, G., & Polycarpou, M. (2019). Parallel ant colony optimization for the electric vehicle routing problem. 2019 IEEE Symposium Series on Computational Intelligence (SSCI).

Mavrovouniotis, M., & Yang, S. (2013). Dynamic vehicle routing: A memetic ant colony optimization approach. In Automated Scheduling and Planning: From Theory to Practice (pp. 283-301). Springer.

Nannen, V., & Eiben, A. E. (2006). A method for parameter calibration and relevance estimation in evolutionary algorithms. Proceedings of the 8th annual conference on Genetic and evolutionary computation.

Ostfeld, A. (2011). Ant colony optimization for water resources systems analysis–review and challenges. Ant colony optimization-Methods and applications.

Parsons, S. (2005). Ant Colony Optimization by Marco Dorigo and Thomas Stützle, MIT Press, 305 pp., $40.00, ISBN 0-262-04219-3. The Knowledge Engineering Review, 20(1), 92-93.

Proctor, S., & Khakoo, M. (2005). Ant colony optimization by Marco Dorigo and Thomas Stützle. Knowl. Eng. Rev., 20(1), 92-93.

Rezvanian, A., Vahidipour, S. M., & Sadollah, A. (2023). An overview of ant colony optimization algorithms for dynamic optimization problems.

Schoonderwoerd, R., Holland, O. E., Bruten, J. L., & Rothkrantz, L. J. (1997). Ant-based load balancing in telecommunications networks. Adaptive behavior, 5(2), 169-207.

Shami, T. M., El-Saleh, A. A., Alswaitti, M., Al-Tashi, Q., Summakieh, M. A., & Mirjalili, S. (2022). Particle swarm optimization: A comprehensive survey. Ieee Access, 10, 10031-10061.

Teodorovic, D., & Dell'Orco, M. (2005). Bee colony optimization–a cooperative learning approach to complex transportation problems. Advanced OR and AI methods in transportation, 51, 60.

Trelea, I. C. (2003). The particle swarm optimization algorithm: convergence analysis and parameter selection. Information processing letters, 85(6), 317-325.

Van den Bergh, F., & Engelbrecht, A. P. (2006). A study of particle swarm optimization particle trajectories. Information sciences, 176(8), 937-971.

Vassiliadis, V., & Dounias, G. (2009). Nature–inspired intelligence: a review of selected methods and applications. International Journal on Artificial Intelligence Tools, 18(04), 487-516.

Wagner, I. A., Lindenbaum, M., & Bruckstein, A. M. (1998). Efficiently searching a graph by a smell-oriented vertex process. Annals of Mathematics and Artificial Intelligence, 24, 211-223.

Yang, L., Wang, Y., & Zhang, J. (2020). Parameter analysis and simulation experiment of ant colony optimization on small-scale tsp problem. IOP Conference Series: Materials Science and Engineering.