Minh Pham

Professor Shamsian

CSCI 270

8 May 2023

Ant Colony Optimization: Its Applications in Transportation and Logistics

Abstract

From the social behaviors of insects and animals, researchers were able to use their behaviors to develop optimization algorithms; those algorithms are swarm intelligence. Ants inspire many methods of swarm intelligence, and it is one of the most popular techniques that we use in swarm intelligence. Ant Colony Optimization (ACO) algorithm is used across many different industries, and advanced versions of ACO were also developed to work with more applications and improve in optimization. Throughout this paper, we will explore the concept of ACO, similar problems to ACO, and its application in transportation and logistics.

Introduction and Background of Ant Colony Optimization

Nowadays, there are many different problems in various industries that use scientific algorithms to solve the problem optimally. As Blum mentioned in the paper, optimization problems are important for both the industrial and scientific worlds since we have limited time and resources. Some notable optimization problems are timetabling, time scheduling, shape optimization, telecommunication network design, and computational biology, just to name a few (2). We must consider some criteria to see if an algorithm is a good option for those optimization problems. Those criteria would be the time it takes to compute an optimal solution with good efficiency and fast computational time. The algorithm must also be versatile since we used it for various cases since different problems will have different sets of input and requirements.

One of the algorithms that can be used for this purpose would be Ant Colony Optimization (ACO); it is a metaheuristic algorithm of approximate solutions for discrete and continuous optimization problems. Since many optimization problems can be highly complex, finding the best solution might be challenging. Using a metaheuristic algorithm would be better since it can provide a solution close to the optimal solution but use much less time and resources. The ACO algorithm was introduced in the early 1990s by Italian computer scientist Marco Dorigo by observing ant colonies. It is one of the most popular optimization techniques we use today. ACO is based on the foraging behavior of ants as they search for food. To mark their favorite paths during their journey, these ants deposit pheromones on the ground as a signal for the ants from the colony to follow this path; the path with more pheromones will correspond to better solutions (Dorigo 1). Since the first introduction of this algorithm, many researchers have developed ACO into different variants and applied it successfully in many applications. We will review the specific applications that use ACO and discover how it compares to different algorithms.

Concepts and Details of Ant Colony Algorithm.

Marco Dorigo describes the overview of this algorithm as "In ACO, a number of artificial ants build solutions to an optimization problem and exchange information on their quality via a communication scheme reminiscent of the one adopted by the real ants" (3). One of the famous problems researchers are trying to solve by using ACO is the Travelling Salesman Problem (TSP), where there is a set of cities with a known distance between them. The goal is to find a Hamiltonian cycle of minimal length on a fully connected graph of the cities. The ACO is reducible to TSP by creating a graph with artificial ants that will move around the graph. Each vertex represents a city for this graph, and each edge connects the cities from two vertices. Each will also have a variable to store the pheromone value, which the ants can modify as they discover the graphs. The core of ACO is an iterative algorithm; we will use a different number of ants for each iteration. At the end of the iteration, the goal is to modify the pheromone values from the previous iteration. The algorithm would influence the future ants to construct a similar solution to the best ones that the algorithm constructed earlier. The author describes the algorithm as the ants building a solution by walking from vertex to vertex on the graph with the constraint of not visiting any vertex that the ants already visited earlier, "an ant selects the following vertex to be visited according to a stochastic mechanism that is biased by the pheromone: when in vertex I, the following vertex is selected stochastically among the previously unvisited ones" (Dorigo 3). The vertex will then be selected with a probability that is proportional to the pheromone associated with the edge between the cities (i,j).

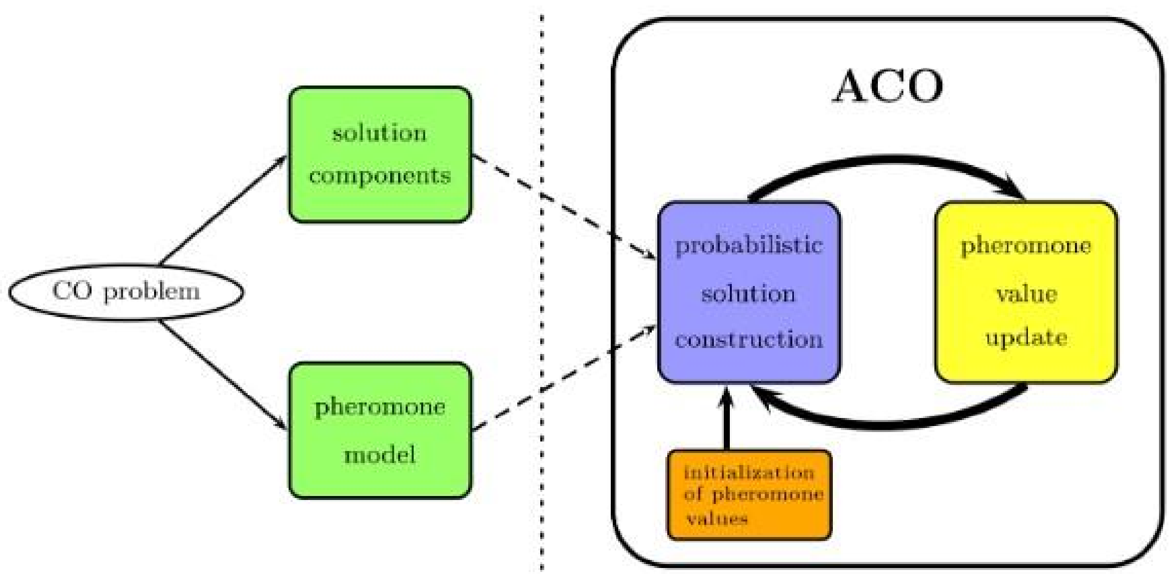


Fig. 4. The working of the ACO metaheuristic.

(Dorigo, Marco, Mauro Birattari, and Thomas Stutzle)

Figure 4 shows how the ACO algorithm works, divided into two steps. The first step would have the candidate solutions constructed with a pheromone model and distributed using probability over the solution space. The second step would be using the candidate solution that we have from step one and using it to modify the pheromone values so that it can bias future sampling toward better quality solutions (Blum 7). The aim of this algorithm is to develop better solutions through iteration with different sizes of ants and use pheromone updates to focus the search on regions of the search space that consist of high-quality solutions. ACO is built from the foundation of reinforcement of solution components based on the quality of the solution. Good solutions comprise many suitable solution components; ACO can create better solutions by learning which components would contribute to a good solution. As the process of solving optimization problems using ACO involves many different steps, it is important to understand the runtime and how the size of the ants and iteration usable can help to improve the runtime for this algorithm. The table below shows the runtime for each step of ACO, where Imax represents the maximum number of iterations. Based on the research result, the runtime for ACO would be O(Imax n^2m) (Li M. 3).

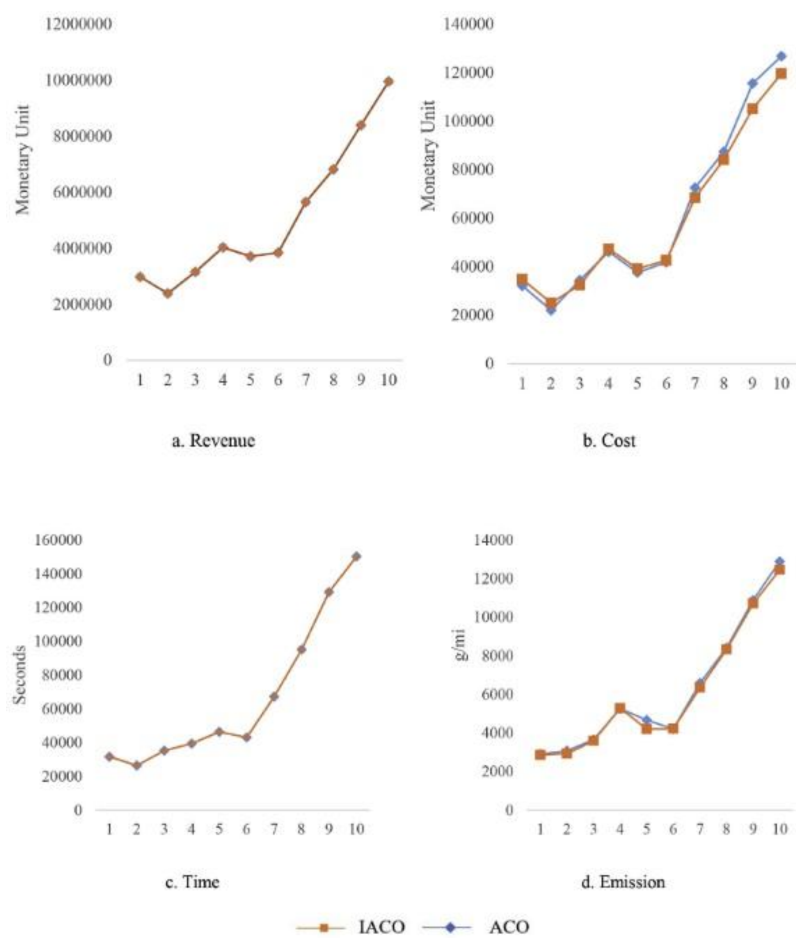
Table 1 Time complexity analysis of ACO algorithm

Step	Description	Complexity of runtime
1	Initialize the situations	$O(n^2 + m)$
2	Set the Tabu-table	$O(n)$
3	Each ant constructs its solution space independently	$O(n^2m)$
4	Evaluate local solutions and calculate the pheromone	$O(n^2m)$
5	Update the pheromone of the trails	$O(n^2)$
6	If $t < I_{\max}$, return to Step 2	$O(nm)$
7	Output the result	$O(1)$

(Li, Munan)

Ant Colony Optimization Application and Advancement in Transportation

In order to optimize the transportation route for vehicles, we have the Vehicle Routing Problem (VRP) that focuses on minimizing costs, time, and emissions for the process of transportation of goods. Researchers use Improved Ant Colony Optimization (IACO) to efficiently solve the problem with better performance compared to the traditional ACO. The goal of the new algorithm is to incorporate a local search procedure and a mutation operator to enhance the searchability of the existing ACO. The algorithm will take in some important constraints, such as the flow balance of the network, the number of vehicles, demand points, product availability, and other factors that impact the transportation system (Li Y. 7).



(Li, Yongbo, Hamed Soleimani, and Mostafa Zohal)

Since the transportation system has multi-depot throughout the network, it is possible to reduce the problem to ACO. The problem can be divided into subproblems where the artificial ants must choose the next node (depot) based on the pheromone level and the distance (Li Y. 7). The process will carry on until the vehicles reach their capacity and return to the depot; the iteration will continue until all the demand points are satisfied, and the level of pheromones is constantly updated until the algorithm reaches the termination point. The researchers also mentioned that IACO delivered good performance with good computational efficiency. The charts above will show some improvements in the improved algorithm compared to the old one. The authors of this paper concluded that "IACO shows a better performance in cost and emission criteria, while the computational time and revenue are similar to ACO." Li Y also added that IACO could produce close-to-optimal results and work with large-scale problems, which will be effective in solving real-world problems (11).

Ant Colony Optimization Application and Advancement in Logistics

We will examine how ACO can be applied to the cold chain logistics distribution industry, where goods must be distributed on time and minimize business costs. Initially, the traditional optimization method requires much time to search the possible options, making it harder to find a close-to-optimal approach, leading to higher costs and poor efficiency. Researchers propose an IACO algorithm that will help to improve the efficiency of cold chain logistics enterprises. The advanced version of ACO for this problem also takes in many different constraints such as transport time factor, transport colling factor, and mean road patency factor in obtaining an optimal result in a large-scale industry. The main difference of the IACO is that the algorithm enhanced the pheromone updating mode by using the constraint function model, which allows one to factor in more variables (Xiong 9).

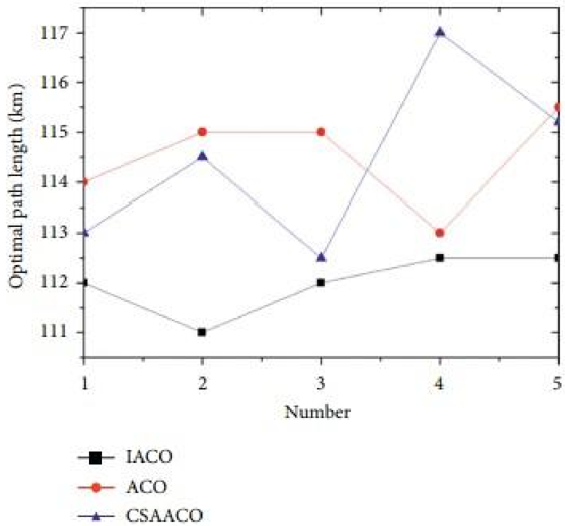


FIGURE 3: The optimal logistics distribution path length comparison.

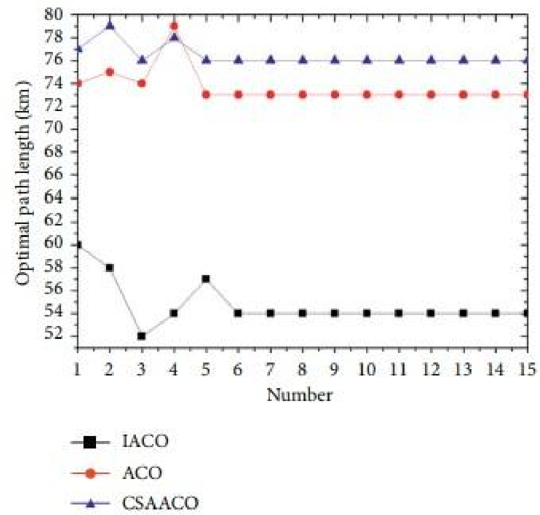


FIGURE 4: Comparison of iteration times of optimal logistics distribution path.

(Xiong, Haiou)

Based on the figures above, the number of iterations that IACO takes to find the optimal logistics distribution path is much less compared to the original version of ACO. When we compared the length of the optimal distribution path, we can also see that the IACO algorithm outperformed the old version of ACO since IACO can compute a shorter optimal path for the logistics chain. There is a notable improvement for IACO because the researcher was able to point out some flaws in the ACO implementation for this problem. ACO uses two pheromone updating strategies: real-time updating and global updating. The defects are that the global update for ACO tends to converge too early, leading to the ants quickly converging on the same initial path so that a better solution cannot be found and obtained (Xiong 5). IACO is focusing on updating the pheromone value based on the ants with the best performance in the operations, hence will create more options for a better solution.

Conclusions

Ant Colony Optimization (ACO) is a metaheuristic algorithm that we can use to solve complex optimization problems in various industries, specifically transportation and logistics, as discussed in the paper. ACO was inspired by the foraging behavior of ants and created solutions iteratively by using artificial ants that move through a graph, deposit, and update the pheromones for each iteration. Some famous problems, such as TSP, could be solved efficiently using ACO. Further improvements of ACO, such as IACO, were able to provide close to the optimal solution with less time and resources, which will significantly benefit our industry needs. Further research and advancements could investigate the scalability of ACO and the benefits it brings to solving larger and more complex optimization problems.

Work Cited

Blum, Christian. "Ant colony optimization: Introduction and recent trends." *Physics of Life reviews* 2.4 (2005): 353–373.

Dorigo, Marco, Mauro Birattari, and Thomas Stutzle. "Ant colony optimization." *IEEE computational intelligence magazine* 1.4 (2006): 28-39.

Li, Munan. "Efficiency improvement of ant colony optimization in solving the moderate LTSP." *Journal of Systems Engineering and Electronics* 26.6 (2015): 1300-1308.

Li, Yongbo, Hamed Soleimani, and Mostafa Zohal. "An improved ant colony optimization algorithm for the multi-depot green vehicle routing problem with multiple objectives." *Journal of cleaner production* 227 (2019): 1161–1172.

Xiong, Haiou. "Research on cold chain logistics distribution route based on ant colony optimization algorithm." *Discrete Dynamics in Nature and Society* 2021 (2021): 1–10.