The Application of Ant Colony Optimization

Zhuobing Zhao

Abstract—**Ant Colony Optimization (ACO) is a powerful metaheuristic inspired by the foraging behavior of real ants, which has been successfully applied to various combinatorial optimization problems. This paper explores the application of ACO to both static and dynamic optimization problems, with a focus on problems such as the Traveling Salesman Problem (TSP), Single Machine Total Weighted Tardiness Scheduling Problem (SMTWTP), Generalized Assignment Problem (GAP), and Set Covering Problem (SCP). We discuss the foundational Ant System (AS) algorithm and its evolution into more sophisticated variants, such as AntNet, which is designed to adapt to dynamic environments, particularly in network routing. Through these examples, we demonstrate the effectiveness of ACO in finding high-quality solutions to complex problems, where both the problem structure and the optimization landscape can change over time. The flexibility of ACO in handling different types of constraints and dynamic changes makes it a highly versatile approach for solving real-world optimization problems.**

I. Introduction

Ant Colony Optimization (ACO) is a recently proposed metaheuristic method used to solve difficult combinatorial optimization problems. The inspiration for ACO comes from the pheromone marking and following behavior of real ants, who use pheromones as a communication medium. Analogous to biological examples, ACO is based on indirect communication among a simple population of agents (called artificial ants), which is achieved through artificial pheromone trails. In ACO, the pheromone trails serve as distributed numerical information, which ants use to probabilistically construct solutions to the problem. During the algorithm execution, ants adaptively adjust these pheromone trails based on their search experiences.

The first example of such an algorithm is the Ant System (AS), which was proposed using the well-known Traveling Salesman Problem (TSP) as an application instance. Although the results were encouraging initially, AS could not compete with state-of-the-art algorithms for TSP. However, it played an important role in stimulating further research, leading to the development of algorithm variants with better computational performance, and it has been widely applied to a variety of problems. In fact, there are now many applications in problems such as quadratic assignment, vehicle routing, sequencing, scheduling, and internet-like network routing that have achieved world-class performance. Motivated by these successes, ACO metaheuristics were proposed as a general framework for existing applications and algorithm variants. From now on, algorithms following the ACO metaheuristic approach will be referred to as ACO algorithms.

The current applications of ACO algorithms can be divided into two important categories: static and dynamic combinatorial optimization problems. Static problems refer to those in which the topological structure and costs do not change during the problem-solving process. For example, the classic Traveling Salesman Problem (TSP) is a static problem where city locations and inter-city distances remain unchanged during the algorithm execution. In contrast, dynamic problems involve changes in the topological structure and costs during the solution construction process. One such example is the routing problem in telecommunication networks, where traffic patterns continuously change.

ACO algorithms for these two categories of problems are very similar at a high level, but there are significant differences in implementation details. The ACO metaheuristic is capable of capturing these differences and is general enough to encompass common ideas shared by both types of applications.

In ACO, artificial ants implement a randomized construction heuristic that makes probabilistic decisions based on artificial pheromone trails and possibly available heuristic information based on input data for the problem to be solved. Therefore, ACO can be interpreted as an extension of traditional construction heuristics, which are readily available for many combinatorial optimization problems. However, the key difference from traditional construction heuristics is that during the algorithm's execution, the pheromone trails are adaptively adjusted based on the accumulated search experience.

II. Application

A. The First ACO Algorithm: Ant System and the TSP

In the Ant System (AS), each ant is initially placed in a randomly selected city and has a memory that stores the partial solution it has constructed (initially, the memory only contains the starting city). Starting from the initial city, the ant gradually moves from one city to another. When the ant is at city i, the probability of ant k choosing to move to an unvisited city j is given by the following formula:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} \quad \text{if} \quad j \in N_i^k \tag{1}$$

where: $\eta_{ij} = \frac{1}{d_{ij}}$ is a priori available heuristic information, with d_{ij} being the distance between cities i and j; α and β are two parameters that determine the relative influence of the pheromone trail and the heuristic information; N_i^k is the set of feasible cities that ant k can choose from when in city i, i.e., the set of cities that ant k has not yet visited.

The parameters α and β influence the algorithm's behavior as follows: If $\alpha = 0$, the selection probability is proportional to $[\eta_{ij}]^\beta$, and the algorithm tends to choose the nearest city. In this case, AS is equivalent to a classic random greedy algorithm (since ants are initially randomly distributed among cities, it is a multi-start algorithm). If $\beta = 0$, only the pheromone reinforcement is effective, which leads to the search quickly entering a stagnation state, and the generated

Algorithm 1 Ant System for TSP

```
Input: Distance matrix $d_{ij}$, number of ants $m$, pheromone evaporation rate $\rho$, number of iterations $T$
1: Initialize pheromone values $\tau_{ij}(0)$
2: for each iteration $t = 1, 2, \ldots, T$ do
3:   for each ant $k = 1, 2, \ldots, m$ do
4:     Place ant $k$ on a random city
5:     Initialize the memory of ant $k$ with the starting city
6:     while ant $k$ has not completed the tour do
7:       for each unvisited city $j$ do
8:         Compute the probability $p_{ij}^k(t)$ using Equation (1)
9:       end for
10:      Select the next city $j$ based on the probability distribution
11:      Add city $j$ to the memory of ant $k$
12:    end while
13:    Evaluate the solution quality of ant $k$ and update pheromone trail $\Delta\tau_{ij}^k(t)$
14:  end for
15:  Update pheromone trails $\tau_{ij}(t+1)$ using Equation (2)
16: end for
Output: Best solution found
```

paths are usually very suboptimal. A stagnation state is defined as a situation where all ants move along the same path and construct the same solution.

The construction of the solution ends after each ant completes its tour, i.e., when each ant constructs a sequence of length n. Afterward, the pheromone trail is updated. In the Ant System (AS), this is done by first reducing the pheromone trail by a constant factor (i.e., pheromone evaporation), and then allowing each ant to add pheromone to the arcs it traversed as follows:

$$\tau_{ij}(t+1) = (1-\rho)\cdot\tau_{ij}(t) + \sum_{k=1}^{m}\Delta\tau_{ij}^k(t) \quad \forall(i,j) \tag{2}$$

where $0 < \rho \leq 1$ is the pheromone evaporation rate, and m is the number of ants. The parameter ρ is used to avoid infinite accumulation of pheromone trails and allows the algorithm to "forget" bad decisions made earlier. For arcs that are not selected by ants, the associated pheromone strength decays exponentially as the number of iterations increases.

$$\Delta\tau_{ij}^k(t) = \begin{cases} \frac{1}{L_k(t)} & \text{if arc } (i,j) \text{ is chosen by ant } k \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $L_k(t)$ is the length of the tour of ant k at time t. According to this formula, the shorter the tour of an ant, the more pheromone it leaves on the arcs it traverses. Generally, arcs chosen by many ants, particularly those that appear in shorter paths, will receive more pheromone and thus be more likely to be chosen in future iterations.

B. The Single Machine Total Weighted Tardiness Scheduling Problem (SMTWTP)

In the Single Machine Total Weighted Tardiness Problem (SMTWTP), n jobs must be processed in sequence on a single machine. Each job has a processing time p_j, a weight w_j, and a due date d_j, and all jobs can start processing at time zero. The tardiness of job j is defined as $T_j = \max\{0, C_j - d_j\}$, where C_j is the completion time of job j in the current job sequence. The goal of SMTWTP is to find a job sequence that minimizes the total weighted tardiness, given by the formula:

$$\sum_{i=1}^{n} w_i \cdot T_i \tag{4}$$

In applying ACO to SMTWTP, the component set C is the set of all jobs. Similar to the TSP case, the state of the problem is all possible partial sequences. In the case of SMTWTP, there are no explicit connection-related costs because the contribution of each job to the objective function depends on the partial solution constructed up to that point.

Algorithm 2 Ant Colony System for SMTWTP

```
1: Input: Processing time matrix $p_j$, weight $w_j$, due date $d_j$, number of ants $m$, pheromone evaporation rate $\rho$, number of iterations $T$
2: Initialize pheromone values $\tau_{ij}(0)$
3: for each iteration $t = 1, 2, \ldots, T$ do
4:   for each ant $k = 1, 2, \ldots, m$ do
5:     Place ant $k$ on an empty sequence
6:     while ant $k$ has not completed the sequence do
7:       for each unscheduled job $j$ do
8:         Compute the probability $p_{ij}^k(t)$ using Equation (4)
9:       end for
10:      Select the next job $j$ based on the probability distribution
11:      Add job $j$ to the memory of ant $k$
12:    end while
13:    Evaluate the solution quality of ant $k$ and update pheromone trail $\Delta\tau_{ij}^k(t)$
14:  end for
15:  Update pheromone trails $\tau_{ij}(t+1)$ using Equation (2)
16: end for
17: Output: Best solution found
```

The ACS algorithm was used to solve the SMTWTP problem (ACS-SMTWTP). In ACS-SMTWTP, each ant starts with an empty sequence and then iteratively adds an unscheduled job to the partial sequence constructed so far. Each ant uses a pseudo-random proportional selection rule to choose the next job, where, at each step, the feasible neighborhood N_i^k of ant k consists of the unscheduled jobs. The definition of pheromone trails is as follows: τ_{ij} represents the priority of scheduling job j at position i. This definition of pheromone trails is widely used in ACO applications to scheduling problems.

Regarding heuristic information, three priority rules were used in to define three types of heuristic information. The studied priority rules include: (i) Earliest Due Date (EDD), which arranges jobs in non-decreasing order of their due dates d_j; (ii) Modified Due Date (MDD), which arranges jobs in non-decreasing order of their modified due dates, given by:

$$\mathrm{mdd}_j = \max\{C + p_j, d_j\} \tag{5}$$

where C is the sum of the processing times of the jobs already scheduled; (iii) Apparent Urgency (AU), which arranges jobs in non-decreasing order of their apparent urgency, given by:

$$\mathrm{au}_j = \frac{w_j}{p_j} \cdot \exp\left(-\frac{\max\{d_j - C_j, 0\}}{k_p}\right) \tag{6}$$

where k is a parameter for the priority rule. In each case, the heuristic information is defined as $\eta_{ij} = \frac{1}{h_j}$, where h_j can be d_j, mdd_j, or au_j, depending on the priority rule used.

The process of global and local pheromone updates is as described in the standard ACS algorithm, where in the global pheromone update, T_{gb} is the total weighted tardiness of the global best solution.

C. Generalized Assignment Problem (GAP)

In the Generalized Assignment Problem (GAP), a set of tasks I, $i = 1, \ldots, n$, must be assigned to a set of agents J, $j = 1, \ldots, m$. Each agent j has a limited capacity a_j, and each task i consumes b_{ij} of the agent's capacity when assigned to agent j. Additionally, the cost d_{ij} of assigning task i to agent j is known. The goal is to find a feasible task assignment that minimizes the total cost.

Let y_{ij} be 1 if task i is assigned to agent j, and 0 otherwise. Then, GAP can be formally defined as:

$$\min f(y) = \sum_{j=1}^{m} \sum_{i=1}^{n} d_{ij} \cdot y_{ij} \tag{7}$$

The constraints are as follows:

$$\sum_{i=1}^{n} b_{ij} \cdot y_{ij} \leq a_j, \quad j = 1, \ldots, m \tag{8}$$

$$\sum_{j=1}^{m} y_{ij} = 1, \quad i = 1, \ldots, n \tag{9}$$

$$y_{ij} \in \{0, 1\}, \quad i, j = 1, \ldots, n \tag{10}$$

Constraint (8) enforces the resource capacity limitations of the agents, while constraints (9) and (10) ensure that each task is assigned to exactly one agent, and no task is assigned to multiple agents.

GAP can easily be transformed into the framework of ACO metaheuristics. The problem can be represented by a graph where the set of components C includes both tasks and agents, i.e., $C = I \cup J$, and the set of connections forms a complete graph. Each assignment solution consists of n pairs of tasks and agents (i, j), similar to how ants traverse the graph. Such a walk must satisfy constraints (9) and (10) to form a valid assignment. A specific way to generate such assignments is through the walk of an ant, which moves from a task node (a node in the set J) to an agent node (a node in the set I), without revisiting any task nodes, but potentially visiting the same agent node multiple times (since multiple tasks can be assigned to the same agent).

In the construction process, each ant must make one of the following two basic decisions : (i) it must decide which task to assign next, (ii) it must decide which agent to assign the selected task to. For both steps, pheromone information and heuristic information can be associated with the tasks. For the first step, pheromone information can be used to learn the appropriate assignment order of tasks, i.e., τ_{ij} gives the attraction of assigning task j immediately after task i. For the second step, the pheromone information is related to the attraction of assigning the task to a specific agent.

To simplify the assumption, we can consider a method where tasks are assigned in a random order. At each step, a task must be assigned to an agent. Intuitively, tasks should be assigned to agents in such a way that the assignment cost is low, and the amount of resources required by the agent is relatively small. Therefore, a possible heuristic information is $\eta_{ij} = \frac{a_j}{d_{ij} \cdot b_{ij}}$, and the AS probability selection rule (7) or the ACS pseudo-random proportional rule can be used for probabilistic selection.

However, one complexity in the construction process for GAP is that it involves resource capacity constraints. In fact, for GAP, there is no guarantee that the solution constructed by an ant is feasible, especially since the solution must satisfy the resource constraint given by (8). To encourage the generation of feasible solutions, the resource constraints must be considered when defining the feasible neighborhood N_i^k for ant k.

For GAP, we define N_i^k as the set of agents to which task i can be assigned without violating the agent's resource capacity. If no agent can satisfy the task's resource requirements, we must construct an infeasible solution. In this case, we can define N_i^k as the set of all agents. Infeasible solutions can then be handled by assigning a penalty based on the amount of resource violation.

D. Set Covering Problem (SCP)

In the Set Covering Problem (SCP), given an $m \times n$ matrix $A = (a_{ij})$, where all elements of the matrix are either zero or one. Additionally, each column has a non-negative cost b_j. If $a_{ij} = 1$, we say column j covers row i. The goal of SCP is to select a subset of columns such that the total cost is minimized and every row is covered. Let J be a subset of columns, and let y_j be a binary variable, where $y_j = 1$ if $j \in J$, otherwise $y_j = 0$. SCP can be formally defined as:

$$\min f(y) = \sum_{j=1}^{n} b_j \cdot y_j \tag{11}$$

The constraints are as follows:

$$\sum_{j=1}^{n} a_{ij} \cdot y_j \geq 1, \quad i = 1, \ldots, m \tag{12}$$

$$y_j \in \{0, 1\}, \quad j = 1, \ldots, n \tag{13}$$

Constraint (12) ensures that each row is covered by at least one column.

ACO can be directly applied to SCP. Columns are the components of the solution, with each column associated with a cost and pheromone trail. The constraint stipulates that each column can be visited by the ants only once, and the final solution must cover all rows. The ants' process of walking on the graph corresponds to gradually adding columns to the current partial solution. Each ant starts with an empty solution and gradually adds columns until all rows are covered.

Each column i has an associated pheromone trail τ_i and heuristic information η_i. The column to be added is chosen with the following probability:

$$p_{ki}(t) = \frac{[\tau_i(t)] \cdot [\eta_i]^\beta}{\sum_{l \in N_k} [\tau_l(t)] \cdot [\eta_l]^\beta} \quad \text{if} \quad i \in N_k \tag{14}$$

where N_k is the feasible neighborhood of ant k, which contains all columns that can cover at least one uncovered row. The heuristic information η_i can be defined in several different ways. For example, a simple static heuristic considering only the column's cost can be used: $\eta_i = 1/b_i$. A more complex approach is to consider the number of rows covered by column i, and define $\eta_i = d_i/b_i$, where d_i is the number of rows covered by column i. The heuristic information can also depend on the partial solution y_k of ant k, in which case it can be defined as $\eta_i = e_i/b_i$, where e_i is the so-called coverage value, which is the additional number of rows covered when column i is added to the current partial solution. In other words, heuristic information measures the unit cost of covering one additional row.

When all rows are covered, the ant finishes constructing the solution. In a post-processing step, the ant may remove redundant columns—those that only cover rows already covered by other columns—or apply additional local search to improve the solution. Pheromone updates can be performed in the standard way as described in previous sections.

When applying ACO to the SCP we have two main differences with the previously presented applications: (i) pheromone trails are associated only to components and, (ii) the length of the ant's walks (corresponding to the lengths of the sequences) may differ among the ants and, hence, the solution construction only ends when all the ants have terminated their corresponding walks.

E. AntNet for Network Routing

In AntNet, ants search for the optimal path from the source node to the destination node in a network graph, and they track the quality of the path using pheromone. The selection of these paths depends not only on the current state of the network but also on the local information and the changing network conditions. The following are the detailed steps and related formulas.

In AntNet, the network can be represented as a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. Each directed edge (i, j) represents a connection in the network. For each directed edge (i, j), a pheromone value τ_{ijd} associated with each destination node d is defined as:

$$\tau_{ijd} \in [0, 1] \tag{15}$$

This represents the pheromone concentration on the edge from node i to node j for the destination node d. Different destination nodes may have different pheromone values on the same path since the ants' goals can vary.

Heuristic information η_{ij} is related to the state of the network link and is typically based on the queue length of the link. Suppose the queue length of link (i, j) is q_{ij}, then the heuristic information can be defined as:

$$\eta_{ij} = 1 - \frac{q_{ij}}{\sum_{l \in N_i} q_{il}} \tag{16}$$

where N_i is the set of neighbors of node i, and q_{ij} is the queue length on link (i, j). The shorter the queue, the higher the heuristic information η_{ij}, indicating that the path is better suited for data transmission.

Each ant starts from the source node s, and its goal is to reach the destination node d. At each step, the ant decides the next node j based on the following probabilistic decision rule:

$$p_{ij}^k(t) = \frac{[\tau_{ijd}(t)] \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{ild}(t)] \cdot [\eta_{il}]^\beta} \tag{17}$$

where: $p_{ij}^k(t)$ is the probability that ant k chooses node j from node i at time t, $\tau_{ijd}(t)$ is the pheromone value on the edge from node i to node j for destination d, η_{ij} is the heuristic information, representing the priority of the link (e.g., queue length), N_i^k is the feasible neighborhood of ant k at node i, consisting of links that can reach an uncovered destination.

β is the weight of the heuristic information, controlling how much influence the heuristic has on the selection. If $\beta = 0$, the decision depends entirely on the pheromone; if β increases, the influence of the heuristic information is strengthened.

When the ant completes its path, it evaluates the path quality, usually measured by the travel time. T_{sd} is the total transmission time from source node s to destination node d. Since the network conditions (such as queue length and traffic) influence the actual delay of the path, the travel time T_{sd} is adjusted based on the current network load.

The quality of the path Q_{sd} can be represented by the following function:

$$Q_{sd} = \frac{1}{T_{sd}} \cdot (\text{local adaptive model}) \tag{18}$$

where the local adaptive model dynamically adjusts based on the network state at each node (e.g., delay, congestion, etc.).

After the ant completes its path search, it updates the pheromone along the nodes it visited. Once it reaches the destination node, the ant returns to the source node along the reverse path, using a high-priority queue to speed up the propagation of information. The pheromone update formula is:

$$\tau_{ijd}(t + 1) = (1 - \rho) \cdot \tau_{ijd}(t) + \Delta\tau_k(t) \tag{19}$$

where: $\tau_{ijd}(t)$ is the pheromone value at time t, ρ is the pheromone evaporation rate, controlling the decay of old information, $\Delta\tau_k(t)$ is the amount of pheromone deposited by ant k, which is proportional to the path quality.

After all ants have completed their path updates, the pheromone on all unused edges (connections) will evaporate according to the following formula:

$$\tau_{ijd}(t) \leftarrow \frac{\tau_{ijd}(t)}{1 + \Delta\tau_k(t)} \tag{20}$$

In AntNet, the quality of the path evaluation is closely related to the current state of the network. Therefore, path selection does not solely depend on delay or bandwidth but also takes into account the dynamic changes in network traffic. This allows AntNet to adaptively select the optimal path in response to varying traffic and network states.

By simulating the behavior of ants searching for the shortest path, AntNet uses dynamically updated pheromones and heuristic information to adapt to real-time changes in the network. It seeks optimal routing paths in packet-switched networks, maintaining high robustness, particularly in networks with time-varying and random conditions.

III. Summary

In summary, we have explored the application of Ant Colony Optimization (ACO) algorithms to a variety of combinatorial optimization problems. Starting from the foundational Ant System (AS) for the Traveling Salesman Problem (TSP), we demonstrated how ACO can be applied to static optimization problems. The Ant System's ability to probabilistically construct solutions based on pheromone trails and heuristic information allows for robust exploration of the solution space.

Additionally, we expanded on the application of ACO to dynamic problems, such as network routing, where network conditions fluctuate in real-time. AntNet, a prominent ACO algorithm for network routing, adapts to these dynamic changes and selects the optimal routing paths by simulating the behavior of real ants searching for the shortest path. This adaptability is essential for maintaining high performance in environments with time-varying network traffic.

We also discussed how ACO has been successfully applied to various other optimization problems, including the Single Machine Total Weighted Tardiness Scheduling Problem (SMTWTP), the Generalized Assignment Problem (GAP), and the Set Covering Problem (SCP). Each of these problems benefits from ACO's ability to handle complex constraints and its flexibility in incorporating both pheromone-based feedback and heuristic information.

Through the various applications discussed, it is clear that ACO is a versatile and powerful metaheuristic for solving combinatorial optimization problems, particularly when the problem environment is dynamic or involves complex constraints. Further research and the development of new ACO variants will likely continue to enhance its performance and expand its applicability to even more diverse real-world problems.