Project 1 Report

Ant Colony Optimization

https://github.com/EathanT/Project1

Specifications:

There is no length requirement for the report. As a general guideline, the report, including pseudocode and figures, will likely be between 3 and 10 pages. It should be as clear and concise as possible while providing a complete, self-contained narrative. Below are some suggested sections for the report along with what they could contain.

- **Introduction -** the introduction may include a brief history of the algorithm along with a description of its purpose and potential applications. This section should address why the algorithm is Important.

- **Intuition -** a high level but complete description of how the algorithm works.

- **Pseudocode and Detailed Description -** formal pseudocode of the algorithm along with a more specific description. While not necessarily a formal proof of correctness, this description should include an argument that the algorithm works as expected. Expectations regarding inputs and outputs should be unambiguous and clear.

- **Run Time Analysis -** a tight asymptotic analysis of the algorithm's run time. Make sure notation and variable meanings are clear. This analysis does not need to consider each line of the algorithm individually but it must address all major components of the algorithm including data structure creation and upkeep as well as any pre- and post-processing steps. This section may also include descriptions and results of different experiments. For example, figures showing

run time as a function of input size or a brief comparison of this algorithm against other approaches to the same problem. Such experiments are not required.

Introduction:

We don't usually collarlate algorithms and living mechanisms with similarities, but nature has a way of figuring out the best way to do things itself. Examples of such are bees using hexagon honey combs, termites with air conditioned mounds, and ants with trail pheromones. Speaking of which: Ants have very unique behaviors in which they use pheromones to create chemical trails that guide other ants from the colony to a food source. This behavior ensures that resources are quickly and efficiently gathered. Ants' brains aren't very sophisticated, but because of this ability, they are able to efficiently navigate and guide their colony: this is known as swarm intelligence. Which brings us to the algorithm I'll be presenting on "Ant Colony Optimization Algorithm", first introduced by Marco Dorigo in the 1990s. This algorithm has been applied to several NP-hard problems like the Traveling Salesman Problem (TSP), Quadratic Assignment Problem (QAP), and Vehicle Routing Problem (VRP). In summary, the importance of ACO lies in its ability to tackle complex optimization problems through a nature-inspired, adaptive, and scalable approach, making it a valuable tool in both theoretical research and practical applications across various disciplines.

Intuition:

Ants, known for their social structures, navigate their environments in fascinating ways. A key principle of the Ant Colony Optimization Algorithm (ACO) is inspired by observing ants as they go from their nests to locate food using the shortest possible paths. Initially, ants move unpredictably, searching for sustenance near their nests, which leads to the exploration of multiple potential routes between the nest and food sources. When ants discover food, they carry portions back to the nest, marking their path with a significant concentration of pheromones. These chemical trails serve as a guide for other ants, influencing their path selection and effectively leading them to the discovered food source. The likelihood of future ants choosing a particular path depends heavily on the pheromone concentration and its rate of

evaporation. Through this observation, we see that the pheromone evaporation rate plays a crucial role, offering insights into the effectiveness of each path, including its length. This natural strategy of path optimization through chemical communication shows a simple yet powerful way to solving complex problems, providing the foundation for the ACO's application in computational tasks. Using this layout of observed behaviors we can make a simple step by step list on how the algorithm will operate:

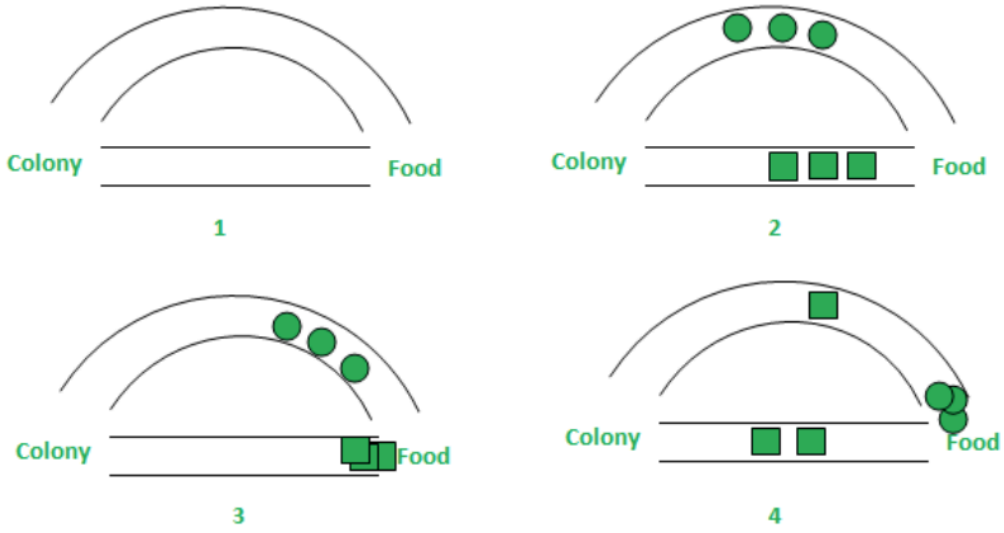


Image: Geeks for Geeks

1. **Stage 1:** All ants are in their nest. There are no pheromones in the environment.
2. **Stage 2:** Ants search with equal(0.5) probability along each path. You can see that the curved path is clearly longer and thus the time taken by ants to reach the food source is greater than the other.
3. **Stage 3:** The ants through the straight path reach their food source earlier. Then they are faced with a similar selection choice, but this time due to pheromone trail along the shorter path already available, probability of selection is higher.
4. **Stage 4:** More ants return via the shorter path and subsequently the pheromone concentrations also increase. Then, due to evaporation, the pheromone concentration in the longer path reduces, decreasing the probability of selection of this path in further stages. Therefore, the whole colony gradually uses the shorter path in higher probabilities. Finally, path optimization is achieved.

Pseudocode and Detailed Description:

Ant colony optimization has been defined into a metaheuristic for connective optimization problems by Dorigo and co-workers**[4].** A metaheuristic is a general set algorithmic framework that can be applied to different optimization problems with relatively few modifications. Which helps identify what we need to change about our algorithm for a set problem.

To use Ant Colony Optimization (ACO) for a problem, you need a framework that includes:

A model *P*=(S,Ω,*f*):

1. **Search Space (S):** A set of decision variables you explore for solutions.

2. **Constraints (Ω):** Rules that solutions must follow.

3. **Objective Function (f):** A function that needs to be minimized.

Key Components:

- **Variables (Xi):** Take values from a set Di, which are all possible values a decision variable can have.

- **Feasible Solution:** A complete assignment of values to variables that satisfies all constraints.

- **Optimal Solution (S*):** The best solution where the objective function is minimized across all feasible solutions.

Pheromone Model in ACO:

Ants build solutions on a graph, G(V, E), where:

- V: Vertices

- E: Edges

Ants move from vertex to vertex, building solutions step-by-step. Pheromones help ants decide on paths:

- They deposit pheromones on components (vertices or edges) they've used.

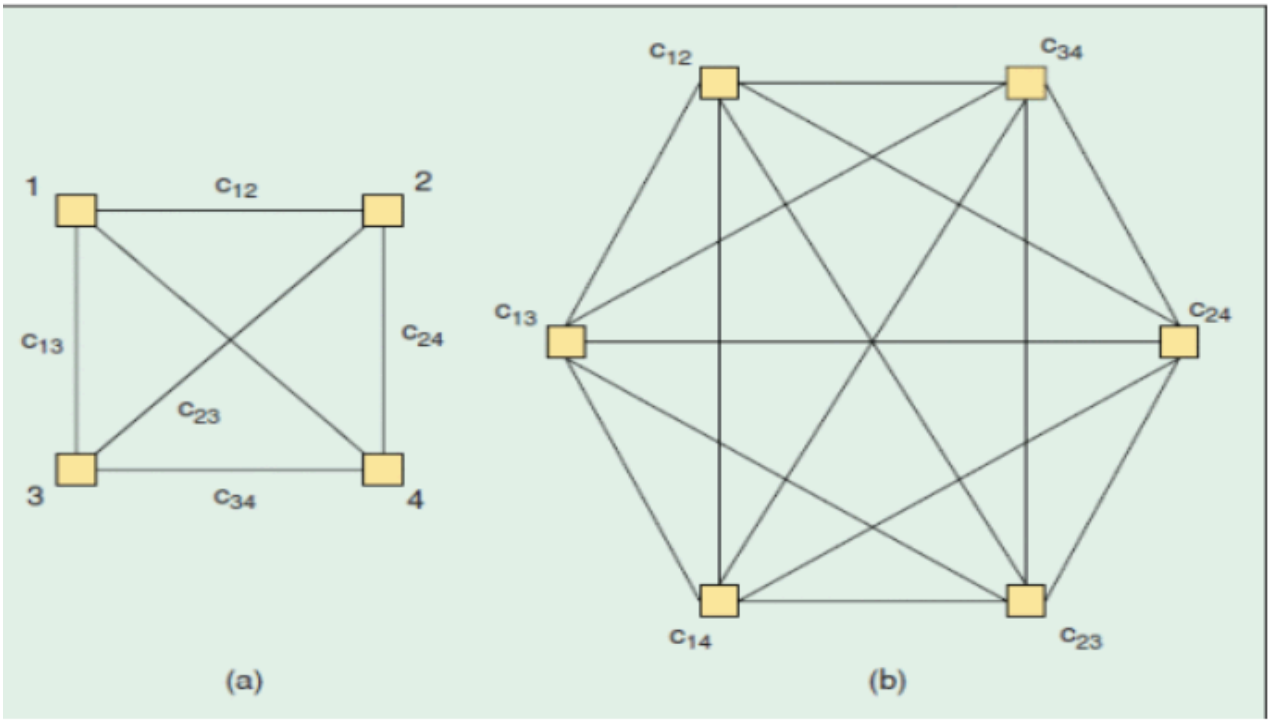
- More pheromones suggest a better solution path for future ants.

These pheromone levels can change based on solution quality, guiding ants toward promising areas in the search space. This approach allows ACO to find optimal or near-optimal solutions efficiently by learning from past search experiences.

Problem:

For this example, we will be using the traveling salesman problem, which has a solution that can be represented by a set of n variables where n is the number of cities. Variable X_i indicates the city to be visited after city i. The solution components here are pairs of cities to be visited one after the other in a given order: *cij = (i,j) Which states that city j, should be visited right after city i. The construction graph is a graph in which the vertices are the cities in the actual traveling salesman problem and the edges are the solution components. GIving us that*

the ants will leave their pheromones on the edges.



[3] Figure 3 Example of possible construction graphs for a four-city TSP where components are associated with (a) the edges or with (b) the vertices of the graph.

The ACO metaheuristic is given by the Algorithm below, After initialization, it iterates after three phases at each iterations:

Algorithm 1 The Ant Colony Optimization Metaheuristic

```
Set parameters, initialize pheromone trails
while termination condition not met do
    ConstructAntSolutions
    ApplyLocalSearch (optional)
    UpdatePheromones
endwhile
```

[3] Algorithm 1

ConstructAntSolutions: A set of ***m*** of ants constructs solutions from elements of a finite set of available solution components $\mathbf{C} = \{c_{ij}\},\ i = 1, \ldots, n,\ j = 1, \ldots, |\mathbf{D}_i|$. A solution construction starts from an empty partial solution $s^p = \emptyset$. At each one of these construction steps, the partial solution s^p is ⌀. . At each construction step, the partial solution s^p is extended by adding a feasible solution component from the set $\mathbf{N}(s^p) \subset \mathbf{C}$, which defined as a set of components that can be added to the current partial solution s^p without violating any of the constraints in Ω. The process of building these solutions can be simplified as a walk on the construction graph *Gc* = (**V,E**). The choice of a solution component from N(s^p) is random probability distribution mechanism, which is biased by the pheromone associated with each of the elements of N(s^p). The rule of the mechanism varies across different ACO algorithms but all of them are inspired by the **[3]** model of the behavior of real ants given in the equation:

$$p_1 = \frac{(m_1 + k)^h}{(m_1 + k)^h + (m_2 + k)^h},$$

that at a given moment in time m1 ants have used the first bridge and m2 the second one, the probability p1 for an ant to choose the first bridge is said equation. where parameters k and h are to be fitted to the experimental data, p2=1−p1. **[4]** Monte Carlo simulations showed a very good fit for k≈20 and h≈2.

Apply LocalSearch: Once solutions have been constructed, and before updating the pheromones, to improve the solutions obtained by the ant, by a local search. This step, which is very specific on the problem at hand, is optional although it is usually included in more advanced ACO algorithms.

UpdatePheromones: The purpose of this step is to increase the pheromone values associated with good or promising solutions, and to decrease the bad ones. Typically, this is achieved by (i) by decreasing all the pheromone values, which we will call pheromone evaporation, and (ii) by increasing the pheromone values that are deemed good solutions.

Used ACO Algorithm:

Several ACO algorithms have been developed over the decades, the original being the Ant System and the other two most successful ones Max-Min Ant System and Ant Colony System. They all have their own benefits and complexity, but for this reports we will just be using the original Ant System(AS) ACO algorithm.

Probability of ant moving from one city to another:

$$p_{ij}^{k} = \begin{cases} \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{c_{ij}\in \mathbf{N}(s^p)} \tau_{il}^{\alpha}\cdot\eta_{1l}^{\beta}} & \text{if } c_{ij} \in \mathbf{N}(s^p), \\ 0 & \text{otherwise,} \end{cases}$$

We will ignore **k** for it just specifies which ant were focusing on,and will also ignore alpha and beta which I'll explain why we need them later, to make things simpler. For the top part of our function:

$$\tau_{ij}^{\alpha}\eta_{ij}^{\beta}$$

- **Tij** = the amount of pheromones between i-city and j-city.
- η_{ij} = the proximity between i-city and j-city

In short this section is just the desire of moving from i-city to j-city.

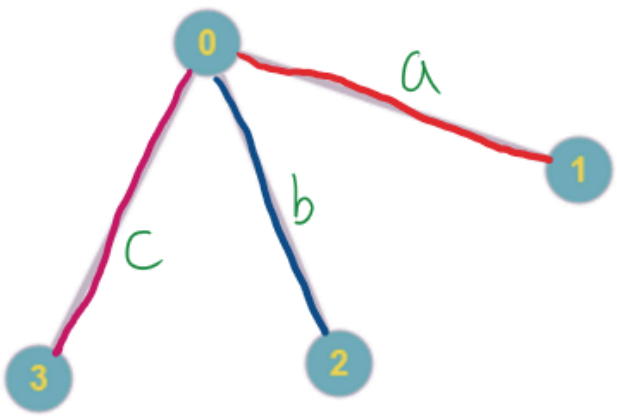
$$\frac{}{\sum_{c_{ij} \in \mathbf{N}(s^p)} \tau_{il}^{\alpha} \cdot \eta_{1l}^{\beta}}$$

- $c_{ij} \in \mathbf{N}(s^p)$ cities in the set of feasible components(that can be reached)
- $\tau_{il}^{\alpha} \cdot \eta_{1l}^{\beta}$ is very similar the top of our function but from cities (i,l) which l is just any other city.

In short this section is the sum of desires of moving from i-city to all other cities that can be reached.

Complete formula: This formula takes the desire of moving from i-city to j-city and divides it by the sum of all other cities that can be reached from i-city (l-city). And gives us the probability of moving from i-city to j-city.

Example of this formula:



$$p_{ij} = \frac{\tau_{ij}^{\alpha} \eta_{ij}^{\beta}}{\sum_{l \in \text{allowed}} \tau_{il}^{\alpha} \eta_{il}^{\beta}}$$

$Citys = \{0, 1, 2, 3\}$

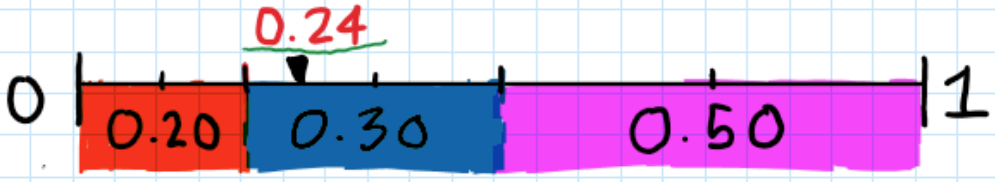
Probablity to moving to each city From 0:

$$\underbrace{\frac{a}{a+b+c}}_{(0,1)} + \underbrace{\frac{b}{a+b+c}}_{(0,2)} + \underbrace{\frac{c}{a+b+c}}_{(0,3)} = \frac{a+b+c}{a+b+c} = \underbrace{1}_{(p_{0,1} + p_{0,2} + p_{0,3})}$$

Example Probalitys of Paths:

$$\underbrace{0.20}_{P(0,1)} + \underbrace{0.30}_{P(0,2)} + \underbrace{0.50}_{P(0,3)} = \underbrace{1}_{P(ALL)}$$

Given a random Number $0 \le x \le 1$: 0.24



Thus since our Random Number Fell into the Domain $0.20 < 0.24 \le 0.30$. City 2 will be moved to.

The Distribution of a Pheromones across the paths:

The pheromone addition to the path city i and city j by an ant k:

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{if ant } k \text{ used edge } (i,j) \text{in its tour} \\ 0 & \text{otherwise,} \end{cases}$$

- $\Delta\tau_{ij}^{k}$ is just the pheromone addition between city i and j by a given ant k
- Q/L_k a constant Q divided by the length of the path L traveled by ant k

In short, this formula determines the amount of pheromones that will be added to the path connecting two cities that were traveled by an specific ant, by dividing a constant by the length of that path.

The amount of pheromones on the path connecting i city and j city.

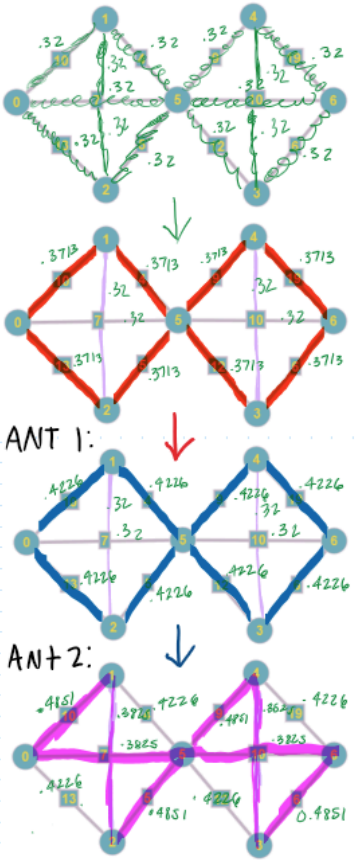
$$\tau_{ij} \leftarrow \rho\tau_{ij} + \sum_{k=1}^{N} \triangle\tau_{ij}^{k},$$

- $\rho\tau_{ij}$ is the current pheromones on the path to city i and city j multiplied the pheromone evaporation coefficient since pheromones are constantly evaporating.
- $\sum_{k=1}^{N} \triangle\tau_{ij}^{k}$ is the sum of all the new pheromones being added that were left by the ants in the path city i and city j.

In short, these formulas will calculate the distribution of pheromones across our paths by finding the pheromone additions between each city using one of our formulas and adding it to pre-existing pheromone value times the pheromone evaporation constant.

Example of these formulas:

An undirected weighted graph:



In this example, lets say that each edge had 0.5 pheromones. And that our pheromone evaporation rate was 64%

$T_{all} = 0.5$

$\rho = .64$ (64% | 36%)

We then have to multiple each path's pheromones by .64: 0.5 x .64 = 0.32 pheromones.

And lets also set Q(constant) to 4(reason later). Now were going to simulate 3 ants, going along each route. Adding to the pheromones each time.

ANT	route	length	Pheromones
0	0,1,2,5,4,6,3,5,2,0	78	$\frac{4}{78} = .0513$
1	0,2,5,3,6,4,5,2,1,0	78	$\frac{4}{78} = .0513$
2	0,5,6,3,4,5,2,1,0	64	$\frac{4}{64} = .0625$

$\Delta T_{ij} = \frac{4}{78} = 0.0513$

$\Delta T_{ij} = \frac{4}{78} = 0.0513$

$\Delta T_{ij} = \frac{4}{64} = .0625$

We can see that since the length of route done by ant 2 is shorter, the amount of pheromones it leaves behind it will be more than any other longer route.

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k \\ 0 \end{cases}$$

$$\tau_{ij} \leftarrow \rho\tau_{ij} + \sum_{k=1}^{N} \Delta\tau_{ij}^{k},$$

Runtime Analysis

To perform an asymptotic runtime analysis of the Ant Colony Optimization (ACO) algorithm, we need to consider the primary components and operations of the algorithm, and their costs, in terms of the problem size (n). In the context of the ACO, this commonly involves the total number of cities (or nodes, (n)) in problems like the Traveling Salesman Problem (TSP).

Primary Components of the ACO Algorithm:

1. **Initialization**:
    - (O(n^2)): Involves initializing pheromones for each edge between the (n) cities. The complexity is quadratic in the number of cities since this requires looping over all possible pairs of cities.
2. **Solution Construction**:
    - (O(m * n^2)): Each of the (m) ants constructs a complete solution (or tour), which involves visiting each node in the graph. For each node, the ant makes a probabilistic choice for the next node based on pheromone levels, which incurs an (O(n)) cost per node, resulting in an (O(n^2)) cost per ant journey.
3. **Local Search (optional)**:
    - (O(m * n^k)): If local search is applied, the complexity depends on the specific heuristic used. A common heuristic is the 2-opt or 3-opt local search, where (k = 2) or (3). This portion is problem-specific and can add significant complexity if used.
4. **Pheromone Update**:
    - (O(n^2)): This involves iterating over all the edges again to update the pheromone levels. Each ant contributes to the pheromone deposition based on the quality of its tour.

5. **Iterations**:

    - Per iteration involves repeating the above steps. The overall complexity will depend on the number of iterations (t) chosen based on a convergence criterion or a fixed limit.

Total Complexity:

Combining these components, the asymptotic runtime complexity per iteration without local search being:

[O(m * n^2 + n^2) = O(m * n^2)]

Including (t) iterations, the overall complexity becomes:

[O(t * m * n^2)]

If local search is included, the complexity per iteration becomes:

[O(m * (n^2 + n^k)) = O(m * n^k)] (since (n^k) dominance when (k >= 2))

Thus, the total complexity over (t) iterations is:

[O(t \times m \times n^k)]

Where:

- (m) = number of ants.
- (n) = number of cities/nodes.
- (t) = number of iterations.
- (k) = degree of local search, usually 2 or 3.

The choice of (m), (t), and whether local search is used significantly influences the performance and runtime of the algorithm in practice.

Resources:

[1]Introduction to Ant Colony Optimization by Geeks for Geeks :
https://www.geeksforgeeks.org/introduction-to-ant-colony-optimization/
[2]Ant colony optimization algorithm by Simulife Hub Video:
https://www.youtube.com/watch?v=u7bQomllcJw
[3]Ant colony optimization | Publisher: IEEE
https://ieeexplore.ieee.org/document/4129846
[4]M. Dorigo and G. Di Caro, "The Ant Colony Optimization meta-heuristic" in New Ideas in Optimization, UK, London:McGraw Hill, pp. 11-32, 1999.