CS201 Project ACO

November 5, 2023

Divyansh Verma (2022CSB1081) ,
Prakhar Maurya (2022CSB1102) ,
Aryan Kumar (2022MCB1259)

Instructor:
Dr. Anil Shukla

Teaching Assistant:
E Harshith Kumar

Summary: In this report, we delve into the world of optimization algorithms, with a primary focus on Ant Colony Optimization (ACO). ACO is a bio-inspired algorithm that draws inspiration from the foraging behavior of ants to solve complex combinatorial problems. To illustrate the practical application of ACO, we have implemented an ACO algorithm in C++ to tackle the well-known Traveling Salesman Problem (TSP). The TSP is a classic NP-hard problem that asks for the shortest possible route that visits a set of cities exactly once and returns to the starting city. By implementing ACO for TSP, we aim to showcase the effectiveness and versatility of this optimization technique in solving real-world problems.

---

1. Introduction

Swarm intelligence is a captivating property observed in systems where the collective behaviors of often simple agents, interacting within their environment, give rise to coherent and functional global patterns. This phenomenon provides a foundation for exploring collective, distributed problem-solving methods, as well as the coordination and establishment of a global model.
For instance:

- A school of fish displays a synchronized swimming pattern as they move in the same direction.
- Ants collaborate in their efforts to locate food sources and efficiently transport them back to the nest.

Swarm intelligence offers valuable insights into how group behaviors can lead to the emergence of sophisticated and effective solutions in various problem-solving scenarios. It's a concept deeply intertwined with optimization techniques like a family of algorithms that fall under Particle Swarm Optimization (PSO).
Particle Swarm Optimization (PSO) is a versatile family of optimization algorithms that takes inspiration from collective and social behavior observed in nature. Within the world of PSO, various algorithms have emerged, each offering a unique approach to solving optimization problems. In this report, we narrow our focus to delve into a specific member of the PSO family, Ant Colony Optimization (ACO).

Ant Colony Optimization (ACO) is a bio-inspired optimization technique introduced by Marco Dorigo in the early 1990s. It has since gained widespread recognition for its ability to solve a variety of combinatorial optimization problems. ACO emulates the way ants find the shortest path between their nest and a food source. This process hinges on the deposition of pheromones and the exploration of paths by individual ants.

In this report, we will explore the foundational concepts, equations, and applications of ACO. Our particular emphasis is on demonstrating how ACO can be effectively employed to solve the Traveling Salesman Problem (TSP), a classical optimization challenge that has long captivated the interest of researchers and practitioners.

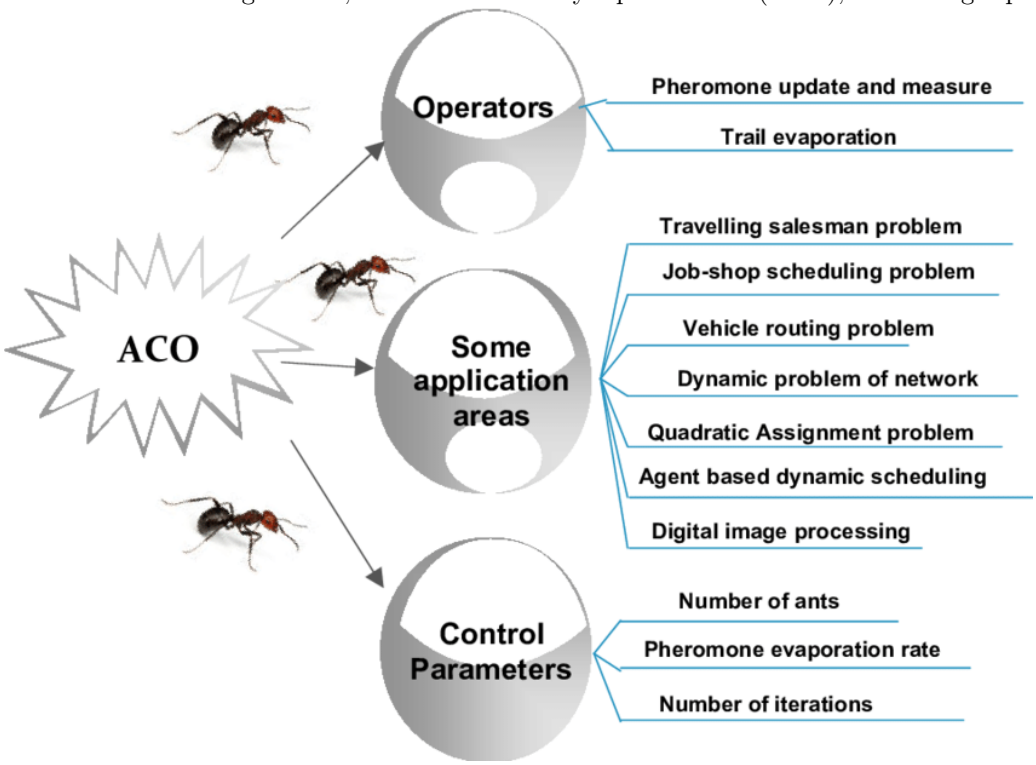
2. The Traveling Salesman Problem (TSP)

2.1. Problem Description :

The Traveling Salesman Problem (TSP) is a classical combinatorial optimization problem that challenges us to find the shortest possible route that visits a set of cities and returns to the starting city. More formally, the problem can be described as follows:

- **Cities**: There are a finite number of cities, often represented as a set, denoted as {C1, C2, C3, ..., Cn}, where 'n' is the total number of cities.
- **Distances**: For each pair of cities, there exists a distance or cost associated with traveling from one city to another. These distances are typically organized in a distance matrix, where the entry at row 'i' and column 'j' represents the distance from city 'Ci' to city 'Cj'.
- **Objective**: The objective of the TSP is to determine the optimal order in which to visit all cities exactly once and return to the starting city, such that the total distance traveled is minimized. The order in which cities are visited is referred to as a tour, and the goal is to find the tour with the shortest total distance.
- **Constraints**: The TSP imposes the constraint that each city must be visited exactly once, and the tour must begin and end at the same starting city.

The TSP is a challenging problem with a wide range of practical applications in real-world scenarios. It is commonly used in logistics, transportation, and route optimization, where minimizing travel distance or time is crucial. Additionally, the TSP has applications in circuit design, manufacturing, and network routing, making it a topic of significant interest in both theoretical and practical optimization research. Solving the TSP optimally becomes increasingly complex as the number of cities grows, leading to the development of various heuristic and metaheuristic algorithms, such as Ant Colony Optimization (ACO), to find high-quality solutions



3. Ant Colony Optimisation (ACO)

3.1. Overview:

Ant Colony Optimization (ACO), introduced by Marco Dorigo in the early 1990s, is a remarkable bio-inspired optimization algorithm. ACO derives its inspiration from the remarkable foraging behavior of ants. This section will provide an overview of ACO and its fundamental concepts, exploring how it mimics the behavior of ants in

finding the shortest path between their nest and a food source.

3.2. Simple Ant Colony Optimisation (SACO)

Ant Colony Optimization (ACO) is a nature-inspired optimization algorithm that draws inspiration from the foraging behavior of ants. It was introduced by Marco Dorigo in the early 1990s and has since gained popularity for solving various combinatorial optimization problems. ACO simulates the way ants find the shortest path between their nest and a food source, a process that relies on the deposition of pheromones and the exploration of paths by individual ants.
In ACO, a population of artificial ants searches for solutions to a given optimization problem by iteratively constructing solutions and adjusting their paths based on local information and a form of communication through pheromone deposition. Over time, paths with higher-quality solutions tend to accumulate more pheromone, guiding other ants to explore those paths.

- **Transition Probability** of selecting the next node $j \in \mathcal{N}$ by the ant k sitting at node i (roulette wheel selection method):

$$P_{ij}^{k}(t) = \begin{cases} \frac{\tau_{ij}^{\alpha}(t)}{\sum_{u \in N_i^k(t)} \tau_{ij}^{\alpha}(t)} & \text{if } j \in N_i^k(t) \\ 0 & \text{otherwise} \end{cases} \quad \text{for } k = 1, 2, 3, \ldots, n_k$$

  where $\mathcal{N}_i^k(t)$ is the set of feasible nodes connected to node i , with respect to the ant k ; $\alpha > 0$ is a constant
- if $\mathcal{N}_i^k(t) = 0$, the predecessor to node i is included in $\mathcal{N}_i^k(t)$.
  - This may cause loops.
  - Loops are Removed when the destination has been reached .

  For Example : **A Loop Found** with path 0-1-3-4-2-3-5-6 then remove the loop and we get the path 0-1-3-5-6

- $x^k(t)$: the solution of ant k, which is a set of nodes visited by ant k.
- $\tilde{x}(t)$: the current best path (the best solution among $x^k(t)$) at generation/iteration t, which is a set of nodes visited by the best ant.
- $\hat{x}(t)$: the global best path found from the first iteration to current iteration of the algorithm.
- $x^+(t)$: the best solution(s) giving the shortest path(s) (the iteration-best or global best ant(s)).
- t: generation/iteration number.
- ρ: evaporation rate.
- n_k: number of ants.
- n_e: number of elite ants.
- (i,j): an edge from node i to node j.
- $\tau_{ij}(t)$: pheromone concentration associated with edge (i,j) at generation/iteration t.
- $\Delta\tau_{ij}^{k}(t)$: the change of pheromone concentration associated with edge (i,j) at generation/iteration t.
- $\Delta\tau_{ij}^{e}(t)$: the change of pheromone concentration associated with edge (i,j) visited by the elite ants at generation/iteration t.
- $f(x^k(t))$: the quality of the solution of ant k.
- $f(\tilde{x}(t))$: the quality of the solution of the $\tilde{x}(t)$ (the best ant).
- $f(x^+(t))$: the cost(s) of the best solution(s) for $x^+(t)$ (the iteration-best or global best ant(s)).
- $L^k(t)$: length of the path (from the source to the destination) constructed by ant k.
- $d_{ij}(t)$: cos between edge (i,j). When t is dropped, d_{ij} is independent of generation/iteration t.
- $p_{ij}^{k}(t)$: transition probability of selecting the next node $j \in \mathcal{N}_i^k(t)$ by ant k and node i.
- $\mathcal{N}_i^k(t)$: the set of feasible nodes connected to node i, with respect to ant k.
- $Q > 0$: a non-zero positive constant.

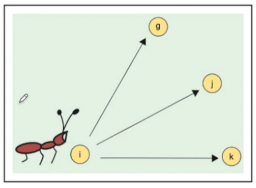
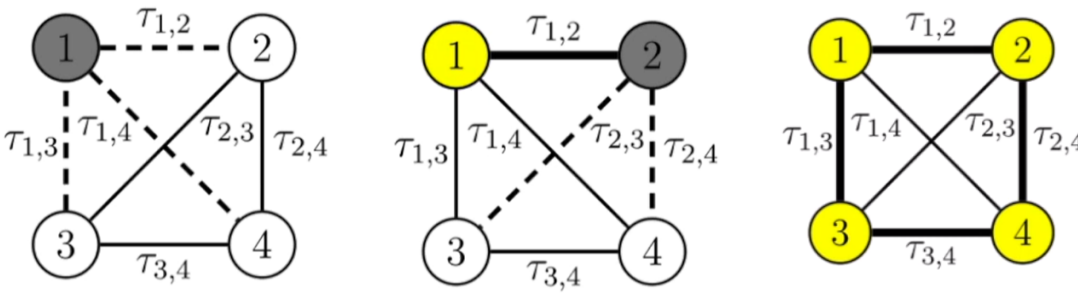
Example 

Table 1: Transition Probabilities for Nodes

Node 1 : $\mathcal{N}_1^k(t) = \{2,3,4\}$	**Node 2 :** $\mathcal{N}_2^k(t) = \{3,4\}$	**Node 4 :** $\mathcal{N}_4^k(t) = \{3\}$
$P_{12}^k(t) = \frac{\tau_{12}^\alpha(t)}{\tau_{12}^\alpha(t)+\tau_{13}^\alpha(t)+\tau_{14}^\alpha(t)}$	$P_{23}^k(t) = \frac{\tau_{23}^\alpha(t)}{\tau_{23}^\alpha(t)+\tau_{24}^\alpha(t)}$	$P_{43}^k(t) = \frac{\tau_{43}^\alpha(t)}{\tau_{43}^\alpha(t)}$
$P_{13}^k(t) = \frac{\tau_{13}^\alpha(t)}{\tau_{12}^\alpha(t)+\tau_{13}^\alpha(t)+\tau_{14}^\alpha(t)}$	$P_{24}^k(t) = \frac{\tau_{24}^\alpha(t)}{\tau_{23}^\alpha(t)+\tau_{24}^\alpha(t)}$	
$P_{14}^k(t) = \frac{\tau_{14}^\alpha(t)}{\tau_{12}^\alpha(t)+\tau_{13}^\alpha(t)+\tau_{14}^\alpha(t)}$		



3.2.1 Evaporation of Pheromone Intensity (Negative Feedback)

Pheromone intensity undergoes evaporation to encourage ants to explore more and prevent premature convergence. This process involves reducing the pheromone intensity on each edge (i, j) according to the evaporation rate $p \in (0, 1)$ (exclusive).
The update of pheromone intensity is performed as follows:

$$T_{ij}^{(t+1)} = (1 - p) \cdot T_{ij}^{(t)}$$

Where:

$T_{ij}^{(t+1)}$: Pheromone intensity on edge (i, j) at time $t + 1$
$T_{ij}^{(t)}$: Pheromone intensity on edge (i, j) at time t
p : Evaporation rate, $p \in (0, 1)$ (exclusive)

The evaporation of pheromone intensity plays a crucial role in the ACO algorithm by preventing the premature convergence of ants to suboptimal solutions and promoting exploration of alternative paths.

3.2.2 Update of Pheromone Intensity (Positive Feedback)

Once all ants have constructed their paths from the source to the destination, and all loops are removed, the pheromone intensity on edge (i, j) is adjusted using a positive feedback mechanism.
The update of pheromone intensity on edge (i, j) is computed as follows:

$$\Delta T_{ij}^{(t)} = \begin{cases} \frac{1}{f(x^*(t))} & \text{if edge } (i,j) \text{ occurs in path } x^*(t) \\ 0 & \text{otherwise} \end{cases}$$

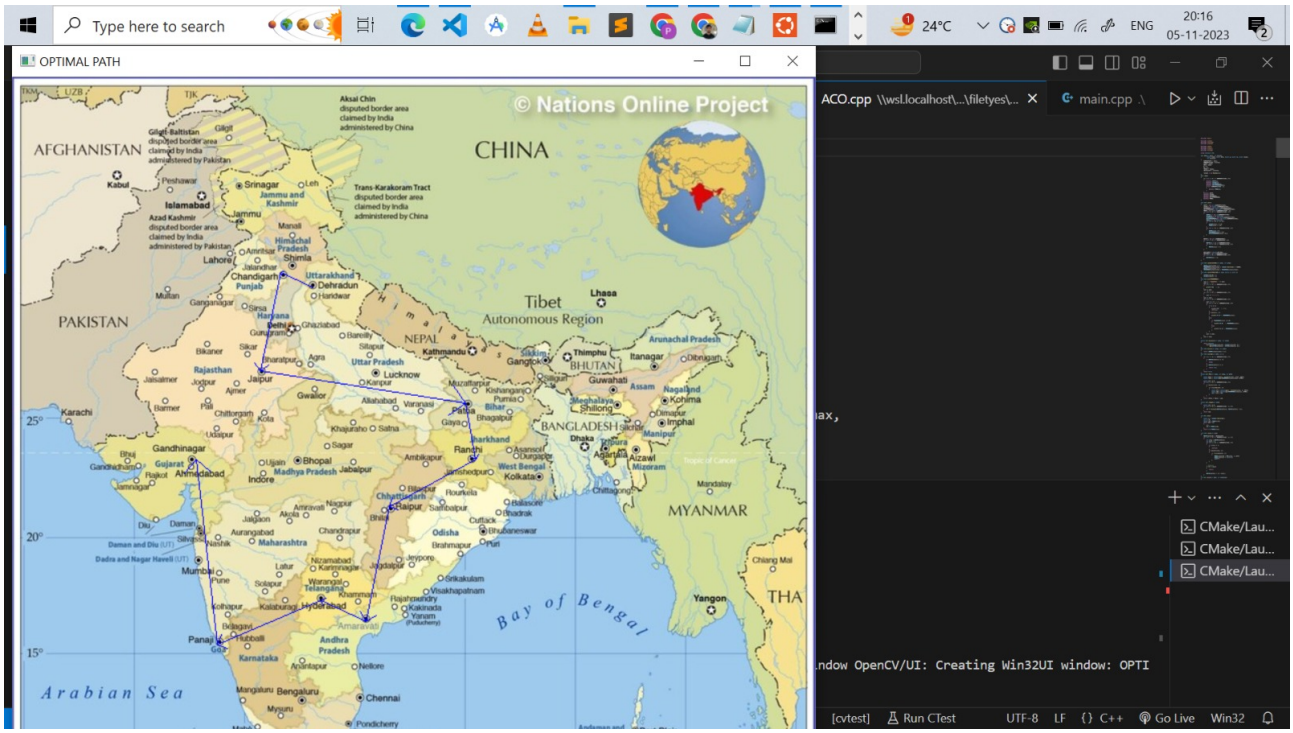
Where:

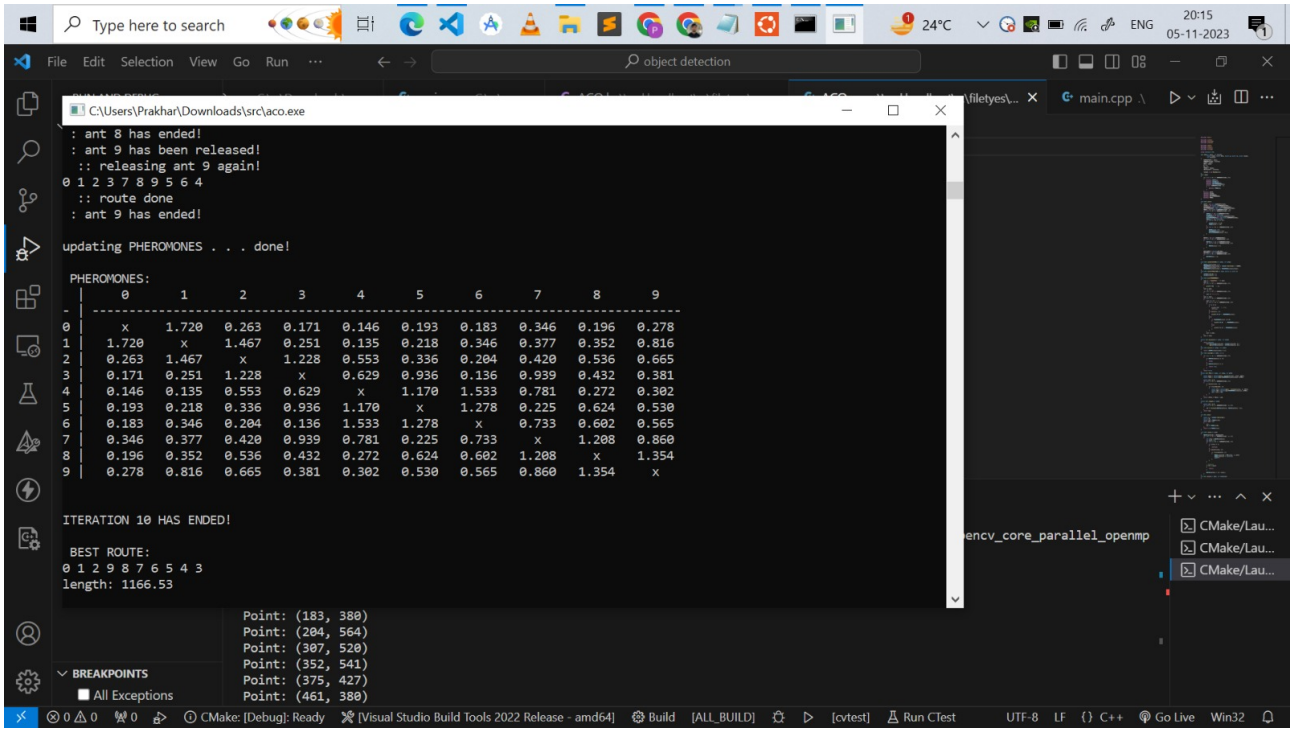
$\Delta T_{ij}^{(t)}$: Change in pheromone intensity on edge (i, j) at time t
$x^*(t)$: The solution of ant k at time t
$f(x^*(t))$: The quality of the solution
α : A constant, where $\alpha > 0$
n_k : The number of ants

The positive feedback mechanism adjusts the pheromone intensity based on the quality of the solutions found by the ants. This reinforcement of pheromone levels on edges that belong to superior paths encourages the exploration of better routes and contributes to the effectiveness of the Ant Colony Optimization algorithm.

4. Interactive Point Selection and ACO Execution

In this section, we illustrate the interactive nature of our ACO implementation. Our code allows users to select points on the map by simply clicking on the screen. These points represent the cities to be visited by the traveling salesman. Once the points are selected, our ACO algorithm is executed, finding an optimized route that visits these cities. The screenshots below showcase the step-by-step process of point selection and the ACO algorithm in action, providing a visual demonstration of our implementation's effectiveness.





4.1. Ant Colony Optimization Algorithm

Algorithm 1 Ant Colony Optimization Algorithm

```
Initialize pheromone levels on all edges
Set parameters: α, β, ρ, Q, t_max
Initialize best solution and best cost
for t = 1 to t_max do
    for each ant k do
        Initialize ant k at a random starting node
        Construct a solution for ant k based on pheromone levels and heuristics
        Update pheromone levels based on ant k's solution
        if ant k found a better solution then
            Update best solution and best cost
        end if
    end for
    Evaporate pheromone levels on all edges
end for
return Best solution found
```

5. Conclusions

In conclusion, our exploration of Ant Colony Optimization (ACO) as a powerful heuristic algorithm for solving the Traveling Salesman Problem (TSP) has provided valuable insights into the world of optimization. ACO, inspired by the foraging behavior of ants, has demonstrated its capacity to tackle complex combinatorial problems and has exhibited promising results in our study.

Through the implementation of an ACO algorithm in C++ to address the classic TSP, we have witnessed its ability to find near-optimal solutions by iteratively refining routes based on pheromone information and heuristic guidance. Our findings underscore the adaptability and robustness of ACO, even in the face of challenging optimization tasks.

While ACO offers numerous advantages, such as its suitability for various real-world applications and its capability to navigate intricate solution spaces, we also acknowledge the challenges it presents. Parameter tuning and sensitivity to problem instances remain areas of concern, emphasizing the need for careful algorithm design and configuration.

This study has contributed to our understanding of ACO's strengths and limitations and has opened doors to future research avenues. As optimization problems continue to grow in complexity and scale, ACO stands as a compelling candidate for addressing these challenges. Further investigations and refinements in ACO algorithms hold the potential for achieving even more remarkable results.

In closing, our report underscores the remarkable potential of ACO as an optimization technique. ACO's adaptability and problem-solving capabilities make it a valuable asset in the toolkit of researchers and practitioners seeking to tackle the most demanding combinatorial optimization challenges.

6. Ant Colony Optimization References

Your report contains a bibliography of references related to Ant Colony Optimization (ACO) and its applications. These references provide the sources consulted during the development of your work.

To cite the ACO references in your manuscript, use the [] command as follows:

- According to Dorigo et al. [**?**], Ant Colony Optimization (ACO) has become a prominent algorithm in the field of computational intelligence.
- Another study by Rai [**?**] applied ACO in the context of edge detection in imagery.
- Manfrin et al. [**?**] explored parallel ACO for solving the Traveling Salesman Problem (TSP).
- Cordon et al. [**?**] introduced the Best-Worst Ant System model, integrating evolutionary computation concepts.