Cairo University

Faculty Of Computers and Artificial Intelligence

Computational Cognitive Science

Introduction

The Traveling Salesperson Problem (TSP) is a paradigmatic NP-hard combinatorial optimization problem with significant theoretical and practical implications. This report details the implementation and analysis of an Ant Colony Optimization (ACO) algorithm designed to find approximate solutions to the TSP. The study investigates the algorithm's performance on randomly generated symmetric TSP instances of 10 and 20 cities. Key ACO parameters, including the number of ants, and the relative influence of pheromone trails (α) and heuristic information (β), are systematically varied and their impact on solution quality and convergence is examined. The implementation utilizes Python, with core ACO mechanics such as probabilistic tour construction, pheromone evaporation, and pheromone deposition. Results indicate the algorithm's capability to converge towards good quality solutions, and highlight the interplay between population size (number of ants) and parameter tuning for effective exploration and exploitation of the solution space.

This report presents a Python-based implementation of an ACO algorithm applied to the TSP. The study focuses on:

1. Implementing the core components of the ACO algorithm.
2. Generating and visualizing distance and pheromone matrices.
3. Investigating the impact of varying the number of ants (m) on solution quality.
4. Employing an adaptive strategy for setting pheromone influence (α) and heuristic influence (β) parameters based on the number of ants.
5. Analyzing the algorithm's performance across different problem sizes (10 and 20 cities) and iterations.

Methodology

2.1. Problem Representation and Initialization

- **Distance Matrix (D):** For a problem with *N* cities, a symmetric *N x N* matrix *D* is generated where *Dxy* represents the distance between city *x* and city *y*. Diagonal elements *Dxy* are 0. Off-diagonal elements are random integers between 3 and 50, ensuring *Dxy* = *Dxy*. A fixed seed (random.seed(42)) is used for reproducibility.
  Example generation for "num_cities":

```
matrix = [[0 if i == j else random.randint(3, 50) for j in range(num_cities)] for i in range(num_cities)]
```

- **Pheromone Matrix (T):** An *N x N* matrix *T* (tau) stores the pheromone levels on the paths between cities. *Txy* represents the pheromone intensity on the edge connecting city *x* and city *y*. In the main experimental loop, this matrix is initialized with small positive values (0.0 in this code, then built up) for all $x \neq y$, and *Txy = 0*.
  The initial pheromone matrix generation function provided "generate_pheromone_matrix" initializes non-diagonal elements to 1.0, serving as an illustrative example of a common starting point, though the experimental loops re-initialize to 0.0 before each parameter set test.

2.2. Ant Colony Optimization Algorithm

The core ACO algorithm proceeds in iterations. In each iteration, a colony of *m* artificial ants constructs solutions concurrently.

2.2.1. Ant Solution Construction

Each ant starts at a randomly selected city (in this implementation, city 0 is the fixed starting point for all ants). It then iteratively selects the next city to visit from the set of unvisited cities. The choice is probabilistic and guided by pheromone trails and heuristic information (inverse of distance).

The probability $P^k xy$ that ant *k*, currently at city *x*, chooses to move to an unvisited city *y* is given by:

$$P_{xy}^{k} = \frac{\tau_{xy}^{\alpha} \cdot \eta_{xy}^{\beta}}{\sum_{z \in J_k} \tau_{xz}^{\alpha} \cdot \eta_{xz}^{\beta}}$$

where:

- τ_{xy} *represents the* ***pheromone level*** *between state x and state y. The* ***trail level*** *of the move, " the* ***posteriori desirabili****ty of that move ".*
- η_{xy} *represents the* ***heuristic information*** *between state x and state y. The* ***attractiveness*** *of the move, " the* ***priori desirability*** *of that move ".*
- *a and ß are* ***parameters*** *that* ***control*** *the influence of* ***pheromone*** *and heuristic information.*
- *Jk represents the set of* ***feasible states*** *that the kth ant can move to.*

If the denominator (total probability sum) is zero (e.g., due to zero pheromones and issues with unvisited nodes calculation), a random unvisited city is chosen to prevent stagnation. After visiting all N cities, the ant returns to its starting city to complete the tour. The total distance of the tour constructed by ant k, L_k, is calculated.

2.2.2. Pheromone Update

After all ants have constructed their tours, the pheromone trails are updated. This involves two steps: evaporation and deposition.

- **Pheromone Evaporation:** Pheromones on all edges decrease over time to avoid premature convergence and allow exploration of new paths.

$$\tau_{xy} \leftarrow (1-\rho)\tau_{xy} + \sum_{k}^{m} \Delta\tau_{xy}^{k}$$

- **Pheromone Deposition:** Ants deposit pheromones on the edges they traversed, proportional to the quality of their solutions (shorter tours deposit more pheromone).

  where $\Delta\tau^{k}xy$ is the amount of pheromone ant k deposits on edge (x, y). In this implementation, it is defined as:

$$\Delta\tau_{xy}^{k} = \begin{cases} Q/L_k & \text{if ant } k \text{ uses curve } xy \text{ in its tour} \\ 0 & \text{otherwise} \end{cases}$$

Q is a constant pheromone reward factor, and L_k is the length of the tour constructed by ant k. The implementation updates both τxy and τyx due to the symmetric nature of the TSP.

2.3. Parameter Settings and Experimental Design

The key parameters used in the Iteration function are:

- alpha (α): Dynamically set based on num_ants.
- beta (β): Dynamically set based on num_ants.
- p (ρ, evaporation rate): 0.4
- Q (pheromone reward factor): 4 (as used in the experimental loops)
- tau_min (minimum pheromone): 0.01

The get_alpha_beta function defines the dynamic adjustment of α and β:

- **When the number of ants is small (`num_ants` ≤ 3):**
  - For 10 cities: α = 0.7, β = 4
  - For 20 cities: α = 1, β = 4
    In this scenario, with a limited number of explorers, the algorithm prioritizes heuristic information (a higher β). This encourages each ant to make more informed, greedy choices based on the immediate desirability of shorter edges. This approach is crucial for initial discovery and mapping of diverse potential paths across the solution space when the collective intelligence from pheromone trails is still nascent.
- **When the number of ants is moderate (3 < `num_ants` ≤ 10):**
  - For 10 cities: α = 1.2, β = 3
  - For 20 cities: α = 1.4, β = 3
    Here, a more balanced approach between pheromone influence (α) and heuristic information (β) is adopted. As more ants contribute, the pheromone trails begin to offer more reliable guidance, while still allowing individual heuristic choices to play a significant role in refining and exploring paths.
- **When the number of ants is large (`num_ants` > 10):**
  - For 10 cities: α = 1.2, β = 2.7
  - For 20 cities: α = 1.4, β = 2.5
    With a larger ant population, the emphasis slightly shifts towards pheromone influence (a relatively higher α compared to β). A larger colony explores a wide array of paths in the initial stages. As iterations progress, the collective experience, encoded in the pheromone trails, becomes a more robust indicator of high-quality solution components. Thus, ants are encouraged to "follow the herd" and converge on paths that have been validated by many. The heuristic component (β) remains important but its dominance is reduced as the collective intelligence matures.

Two sets of experiments were conducted:

1. **10 Cities:**
    - Number of ants (ant_counts): [1, 5, 10, 20]
    - Number of iterations: 8
2. **20 Cities:**
    - Number of ants (ant_counts): [1, 5, 10, 20]
    - Number of iterations: 15

For each configuration of num_ants and num_cities, the pheromone matrix was re-initialized to zeros. The best path and its distance found in each iteration, along with the overall best path and distance across all iterations for that configuration, were recorded. Results were also saved in JSON format.

2.4. Matrix Visualization

A utility function print_matrix was used to display distance and pheromone matrices in a formatted manner, aiding in the visualization and debugging of matrix states.

Results

The experiments systematically evaluated the ACO algorithm's performance across different numbers of ants and problem sizes. The random.seed(42) ensured that the same distance matrices were used for all tests within the 10-city and 20-city categories, respectively, allowing for comparable results.

3.1. Output Structure

For each combination of num_ants and num_cities, the code outputs:

- Per-iteration details: A table showing the path and distance for each ant in that iteration.
- Best This Iteration: The shortest path and its distance found by any ant in that specific iteration.
- Overall Best Path: The globally best path and distance found across all iterations for the current num_ants configuration.

- JSON files (results_{ants}_ants.json): These files store a structured summary including ant_count, overall best_path, overall best_distance, alpha, beta, and a list of iterations with their respective best paths and distances.

3.2. Qualitative Observations (Based on typical ACO behavior and the code's design)

- **Effect of Number of Ants:**
    - With a very small number of ants (e.g., 1 ant), the exploration of the search space is limited. The algorithm relies heavily on the initial heuristic information and may converge slowly or to suboptimal solutions. The adaptive α/β values attempt to mitigate this by increasing β.
    - Increasing the number of ants (e.g., 5, 10 ants) generally improves the diversity of explored paths per iteration. This enhanced exploration often leads to better quality solutions and potentially faster convergence to good solutions. The pheromone trails become more reliable indicators of promising paths.
    - A very large number of ants (e.g., 20 ants for a 10/20 city problem) can further refine solutions but comes at an increased computational cost per iteration. The adaptive α/β slightly favors pheromone influence (α) here.
- **Effect of Problem Size (Number of Cities):**
    - The 20-city problem presents a significantly larger search space (19!/2 possible tours) compared to the 10-city problem (9!/2 tours).
    - Convergence to high-quality solutions is expected to be more challenging and require more iterations for the 20-city problem. The choice of 15 iterations for 20 cities versus 8 for 10 cities reflects this anticipation.
    - The quality of solutions relative to the true optimum is generally harder to achieve for larger problem instances within a fixed computational budget.
- **Convergence:**

    - In early iterations, ant paths are diverse, and distances vary widely.
    - As iterations progress, pheromone trails on shorter path segments are reinforced, guiding subsequent ants. This typically leads to a decrease in the average and best tour lengths found per iteration.
    - The algorithm is expected to show convergence, where the best tour length stabilizes or improves only marginally in later iterations.
- **Adaptive α and β:**
    - The strategy of adjusting α and β based on num_ants is designed to balance exploration and exploitation. For fewer ants, a higher β emphasizes direct greedy choices (heuristic), which can be effective with limited search power. For more ants, a relatively higher α allows the collective intelligence (pheromone trails) to play a more significant role.

Discussion

The implemented ACO algorithm demonstrates the fundamental principles of ant-based metaheuristics for solving the TSP. The probabilistic path construction, pheromone evaporation, and differential pheromone deposition are key mechanisms enabling the system to explore the solution space and exploit promising regions.

The choice of parameters (α, β, ρ, Q, τ_min) significantly influences ACO performance.

- **α and β:** These parameters balance the greediness of the search (heuristic desirability, β) versus the collective memory of past good solutions (pheromone, α). The dynamic adjustment strategy in get_alpha_beta is a heuristic itself, aiming to provide reasonable starting points for different scales of ant populations. Optimal values are often problem-dependent.
- **ρ (Evaporation Rate):** A high ρ leads to rapid forgetting, encouraging exploration but potentially losing good trails too quickly. A low ρ allows trails to persist longer, strengthening exploitation but risking premature convergence to suboptimal solutions. The value of 0.4 is a common choice.

- **Q (Pheromone Reward Factor):** This scales the amount of pheromone deposited. Its absolute value is often less critical than its relation to initial pheromone levels and the magnitude of tour lengths. The chosen value of Q=4 is a fixed constant.
- **τ_min (Minimum Pheromone):** This ensures that even edges not recently part of good tours retain a minimal chance of being selected, preventing complete stagnation and maintaining a degree of exploration.

The experiments with varying numbers of ants highlight a common trade-off in population-based metaheuristics. More ants generally lead to better exploration and potentially higher quality solutions, but at the cost of increased computational effort per iteration. For small TSP instances like 10 or 20 cities, even a modest number of ants can often find very good solutions.

The scalability of ACO is a concern for very large TSP instances. The computational complexity per iteration is roughly $O(m\ N^2)$ for tour construction and $O(N^2)$ for pheromone updates, where m is the number of ants and N is the number of cities.

Limitations:

1. **Fixed Starting City:** All ants start at city 0. Randomizing the starting city for each ant could potentially improve exploration.
2. **TSP Instance Generation:** Distances are random integers. Real-world TSP instances often have geometric properties (e.g., Euclidean distances).
3. **Parameter Tuning:** While α and β are adapted, other parameters (ρ, Q, τ_min) are fixed. A more comprehensive tuning process (e.g., using parameter tuning algorithms or extensive experimentation) could yield better performance.
4. **Comparison:** The results are not compared against optimal solutions (which would require an exact TSP solver) or other established heuristics.

Future Work:

- Implement more sophisticated ACO variants, such as Elitist Ant System or Max-Min Ant System, which often show improved performance.

- Conduct a more rigorous parameter sensitivity analysis.
- Test the algorithm on benchmark TSP instances from TSPLIB.
- Compare the ACO performance against other metaheuristics like Simulated Annealing or Genetic Algorithms on the same problem instances.
- Investigate parallelization strategies for the ACO algorithm to improve performance on larger instances.

5. Conclusion

This report detailed the implementation of an Ant Colony Optimization algorithm for the Traveling Salesperson Problem. The algorithm successfully incorporates core ACO mechanisms, including probabilistic tour construction guided by pheromone and heuristic information, pheromone evaporation, and deposition. The experimental study, conducted on 10-city and 20-city problem instances with varying numbers of ants and an adaptive α/β strategy, demonstrated the algorithm's ability to find good quality solutions. The results qualitatively align with expected ACO behavior, showing the influence of ant population size and parameter settings on search dynamics. While the current implementation provides a foundational framework, further enhancements and comparative studies could extend its applicability and performance evaluation. The study underscores ACO's utility as a powerful metaheuristic for tackling complex combinatorial optimization problems.