Evaluating Performance and Optimality of Ant Colony Optimization and Christofides Algorithm in Approximating the Traveling Salesman Problem.

Shirish Mankal
Computer Science
Portland State University, Portland, Oregon
Email: mankal@pdx.edu

1. Abstract:

The Traveling Salesman Problem (TSP) presents a classic combinatorial optimization challenge that has been extensively studied in the field of computer science and operations research. This project investigates the efficacy of two distinct algorithms in solving the TSP: Ant Colony Optimization (ACO), a probabilistic technique inspired by the foraging behavior of ants, and the Christofides algorithm, a deterministic approach that guarantees a solution within a 1.5-factor of the optimal path length. Through rigorous experimentation, both algorithms were subjected to a variety of input conditions to assess their relative efficiencies and accuracies. The results highlight ACO's proficiency in generating high-quality solutions, particularly excelling with incrementally complex inputs. Christofides Algorithm distinguished itself with its rapid computation, asserting its advantage in simpler, well-structured scenarios. The study confirms that while ACO approaches the optimal with impressive consistency, Christofides swift computations make it an invaluable tool when time is of the essence. Collectively, the findings offer insightful implications for algorithm selection in solving TSPs, enriching the toolbox for practitioners seeking tailored optimization strategies.

Key-words: Traveling Salesman Problem, Ant Colony Optimization, Christofides Algorithm, Algorithm Performance, Approximation Algorithms

Source link: https://github.com/shirish2001/Algo_Final_Project

2. Introduction:

Given a collection of cities and the distance of travel between each pair of them, the traveling salesman problem is to find the shortest way of visiting all of the cities and returning to the starting point. Though the statement is simple to state but it is more difficult to solve. Traveling Salesman Problem is an optimization problem and has a vast search space and is said to be NP-hard, which means it cannot be solved in polynomial time. It is one of the most fundamental problems in the field computer science in today's time. The Traveling salesman problem is being applied in many fields nowadays. Some of its applications are vehicle routing, manufacturing of microchips, packet routing in GSM, drilling in printed circuit boards etc. In simpler words, say we have a set of n number of cities, and then we can obtain (n - 1) and alternative routes for covering all the cities. Traveling salesman problem is to procure the route which has the least distance.

This complexity is the impetus for my investigation into efficient algorithms that provide feasible solutions to the TSP. In this study, I compare two distinctive methodologies: the Ant Colony Optimization (ACO) algorithm, inspired by the foraging behavior of ants, and the Christofides algorithm, noted for its mathematical guarantee in providing a solution that comes close to the optimal path.

I chose the ACO and Christofides algorithms for their contrasting approaches: the ACO algorithm is lauded for its adaptability and heuristic nature, while the Christofides algorithm is deterministic, offering a proven guarantee for achieving a solution within a certain threshold of the shortest possible path. The purpose of my study is to understand how these algorithms perform across a variety of TSP instances and determine their efficacy in approximating the optimal solution.

This report details the journey from theoretical exploration to practical analysis of algorithms for solving the Traveling Salesman Problem (TSP). It includes the mechanics of the algorithms, the results of rigorous testing across different TSP difficulties, and the learned lessons. The results direct the choice of suitable heuristics for various scenarios, demonstrating the advantages and suitability of every strategy in dealing with this traditional optimization problem.

3. Algorithms Description

3.1 Ant Algorithm:

The Ant Colony Optimization (ACO) algorithm, inspired by the natural foraging behavior of ants and introduced in the early 1990s by Marco Dorigo [1], is a probabilistic technique designed to solve computational problems that can be represented as finding optimal paths through graphs. This algorithm takes inspiration from the fact that ants can find the quickest route between their nest and food sources by leaving behind and following a chemical trail called a pheromone [2]. At each stage, the ant chooses to move from one city to another according to some rules:

1. It is necessary to visit each city exactly once.
2. Cities farther away are less likely to be selected due to lower visibility.
3. A city that has a stronger pheromone trail between its edges is more likely to be chosen.
4. An ant, especially one that traveled a shorter distance, leaves more pheromones on each path it traversed after completing its circle.
5. After each iteration, pheromone trails evaporate.

3.2 Pseudocode:

```
input: cities, alpha, beta, n_ants, n_iterations, rho, pheromone_deposit
output: best_tour
1. Calculate Euclidean distance for the cities and get the number of cities as citynum
2. Initialize pheromone on all paths with tau_init
3. for iteration from 1 to n_iterations do
4. Initialize an empty list all_tours
5. for ant from 1 to n_ants do
6. tour <- construct_path (cities, pheromone, alpha, beta)
7. tour <- apply_2_opt (tour, distance_matrix)
8. tour_distance <- calculate_path_length (tour, distance_matrix)
9. Append (tour, tour_distance) to all_tours
10. end for
11. Update pheromone levels:
12. for each path do
13. pheromone[path] *= (1 - rho)
14. end for
15. for (tour, tour_distance) in all_tours do
16. for i from 0 to citynum - 2 do
```

17. Update pheromone on the edge (tour[i], tour[i+1]): pheromone[tour[i], tour[i+1]] += pheromone_deposit / tour_distance
18. end for
19. end for
20. end for
21. Select the best_tour from all_tours based on the shortest tour_distance
22. Return best_tour

3.3 ACO Algorithm Adaptations and Contributions for TSP:

Adaptation	Description	Contribution
1. Uniform Pheromone Initialization	Every path begins with the same pheromone levels.	Encourages exploration and avoids early biases.
2. Local and Global Pheromone Updates	Pheromones are updated individually and collectively after all ants' tours	Balances individual learning with group intelligence, refining the search process.
3. Roulette Wheel Selection	Using a probabilistic approach, pheromone strength and heuristic data are combined to determine a path.	manages the trade-off between using well-known paths and discovering new ones.
4. 2-Opt Local Search	Applies local optimizations to improve individual ant tours.	Enhances solution quality by removing path inefficiencies.
5. Pheromone Evaporation	Reduces pheromones concentration over time.	keeps discovery from stopping too soon and promotes it continuously.
6. Fixed Parameter Setting	Parameters α, β, and ρ are set prior to execution.	Simplifies performance analysis by providing consistent operating conditions.
7. Comparative Analysis with NetworkX	Implements Christofides' algorithm for benchmarking.	provides a performance comparison with another well-known technique for solving TSPs.
8. Execution Time Measurement	Captures the time taken to complete the algorithm's execution.	Provides insights into computational efficiency and complexity.
9. Solution Visualization	Graphically depicts the best tours found by ACO.	proves the ACO's ability to solve real-world problems and supports qualitative assessment.

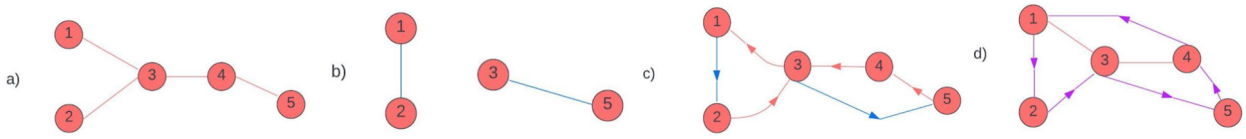
3.4 Christofides Algorithm:

Christofides' Algorithm, introduced in 1976 by Nicos Christofides, provides a near-optimal solution to the Traveling Salesman Problem (TSP) [3], ensuring the solution is no more than 50% longer than the shortest possible route. By starting with a Minimum Spanning Tree (MST) to connect all cities minimally, it expertly integrates ideas from graph theory. After that, the algorithm uses Minimum Weight Perfect Matching (MWPM) to address odd-degree vertices in order to create a Eulerian circuit. This circuit is then condensed into a Hamiltonian cycle that makes exactly one visit to each city [4]. This method is especially important for the NP-hard problem TSP, since it provides a guaranteed performance bound along with a

computationally feasible solution. Because it strikes a balance between efficiency and solution quality, Christofides' Algorithm continues to be a mainstay in heuristic optimization. The same strategy can be carried out to yield a 3/2-approximation algorithm:

1. Find a minimum spanning tree T.
2. Let O be the set of nodes with odd degree in T.
3. Find a minimum-cost perfect matching M on O.
4. Add the set of edges of M to T.
5. Find a Eulerian tour.
6. Shortcut the Eulerian tour.

The below figure illustrates the christofides algorithm on a simple five-city instance of TSP:



3.4.1 Theoretical Underpinning:

Christofides algorithm is grounded in graph theory and combines several sophisticated concepts to form its heuristic:

Component	Description	Role in Christofides Algorithm
1. Minimum Spanning Tree (MST)	A subset of edges that connects all vertices with no cycles and minimum possible total edge weight.	Forms the backbone of the solution, ensuring all cities are connected in the most efficient manner.
2. Matching	Identifies and pairs vertices of odd degree within the MST to add the least possible weight.	Ensures all vertices have an even degree, a prerequisite for forming an Eulerian circuit.
3. Eulerian Circuit	A path that visits every edge exactly once and returns to the start, requiring all vertices to have an even degree.	Utilizes the modified MST to find a path that covers all connections while minimizing travel cost.
4. Shortcuts	Skips repeated cities in the Eulerian circuit to avoid visiting any city more than once.	Converts the Eulerian circuit into a Hamiltonian circuit, fulfilling the TSP requirement of visiting each city exactly once.

3.4.2 Pseudocode:

input: cities
output: best_tour
1. Calculate Euclidean distance for the cities and get the number of cities as citynum
2. G <- create_complete_graph (cities, distance_matrix)
3. T <- find_minimum_spanning_tree(G)
4. O <- find_nodes_with_odd_degree(T)
5. M <- find_minimum_weight_perfect_matching (G, O)

6. multigraph <- combine (T, M)
7. euler_tour <- find_eulerian_circuit(multigraph)
8. best_tour <- convert_eulerian_to_hamiltonian(euler_tour)
9. best_tour_distance < calculate_path_length (best_tour, distance_matrix)
10. Return best_tour

---

4. Methodology

4.1 Experimental Procedure:

The experimental procedure for evaluating the Ant Colony Optimization (ACO) algorithm and Christofides algorithm on the Traveling Salesman Problem (TSP) was meticulously designed to ensure a comprehensive assessment of both algorithms' performance and efficacy. Below is a detailed description of how the algorithms were implemented and executed:

4.2 Initialization:

For both ACO and Christofides algorithms, a set of n cities was represented as nodes in a graph, with their coordinates (coord_x, coord_y) determining the distances (distance_matrix) between each pair of cities using the Euclidean distance formula.

a) ACO Algorithm Implementation:

1. **ACO Setup:**
   - A population of 'm' ants was generated, each tasked with constructing a solution to the TSP by traversing the graph.
   - To promote exploration, pheromone levels ('pheromone_matrix') were initialized on all pathways. parameters like 'beta' (heuristic desirability) and 'alpha' (pheromone influence), 'evaporation_rate', and 'pheromone_deposit' were set based on preliminary tests to optimize performance

2. **Solution Construction:**
   - In order to ensure that distant cities are less likely to be selected, each ant built a tour by selecting the next city using a probabilistic formula that takes into consideration the strength of the pheromone trail and the heuristic value (inverse of distance).
   - This decision-making process was made easier by the 'roulette_selection' strategy, where ants preferred pathways with higher pheromone levels.

3. **Optimization and Pheromone Update:**
   - Upon completing their tours, ants applied a 2-opt local optimization to enhance the solution quality.
   - Then, extra pheromones were added to shorter excursions in order to reinforce the paths that proved successful. Additionally, a method for evaporation was used to avoid pheromone saturation**.**

b) Christofides Algorithm Implementation:

1. **Minimum Spanning Tree (MST):**

- Utilizing the NetworkX library, a Minimum Spanning Tree (MST) was derived from the graph. The MST ensured a subgraph connecting all vertices (cities) with the minimum total edge weight, laying the groundwork for an efficient preliminary tour.

2. **Minimum Weight Perfect Matching (MWPM) on Odd Degree Vertices:**
    - The algorithm identified vertices with odd degrees inside the MST. The next step was to construct a subgraph with these odd-degree vertices, and to determine the least costly way to couple these vertices, a Minimum Weight Perfect Matching (MWPM) was carried out. In order to build an Eulerian circuit, this step is essential.

3. **Formation of Eulerian Circuit and Conversion to Hamiltonian Cycle:**
    - By adding the matched edges to the MST, we ensured all vertices had even degrees, allowing for the formation of a Eulerian circuit using NetworkX's functionalities. This circuit makes at least one pass around each edge. We converted this Eulerian circuit into a Hamiltonian cycle by shortcutting in order to avoid making several trips to the same city in order to comply with TSP requirements, which state that each city must be visited precisely once.

Experimental Execution:

- The experiments were conducted over a predefined number of iterations (iterations), with each iteration representing a complete execution of the ACO and a single run of Christofides algorithm for comparative analysis.
- Execution time was measured for both algorithms to assess their computational efficiency.
- The best path and its length were recorded and visualized using matplotlib, showcasing the algorithms' ability to find near-optimal solutions to the TSP.
- The results were analyzed to compare the performance of the ACO against the theoretical guarantee offered by Christofides algorithm, providing insights into their respective strengths and limitations in solving the TSP.

4.3 Key Measures:

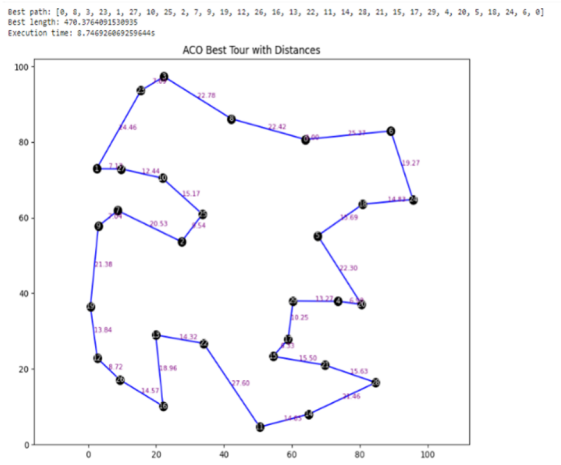
To ensure an unbiased comparison between the Ant Colony Optimization (ACO) algorithm and Christofides algorithm for solving the TSP, implemented the following measures:

- **Consistent Environment:** To avoid performance variability, all testing were carried out in a dedicated computing environment with set hardware requirements.
- **Minimized System Load:** Background processes were minimized to ensure maximum resource availability for the experiments.
- **Garbage Collection Control:** In order to prevent interference and keep a consistent memory state between tests, managed garbage collection was implemented.
- **Identical Test Instances:** To ensure fairness, tests of both algorithms were conducted using identical TSP instances.
- **Iteration Uniformity:** Applied the same number of iterations for each algorithm to provide equal optimization opportunities.
- **Parameter Optimization (ACO):** In order to remove algorithm performance variance caused by parameter settings, the ACO's parameters were improved and fixed before testing.

- **Statistical Validity:** Employed statistical analysis to validate the performance differences, ensuring observed variations were significant.

5. Experimental Results

Ant Colony Optimization (ACO):

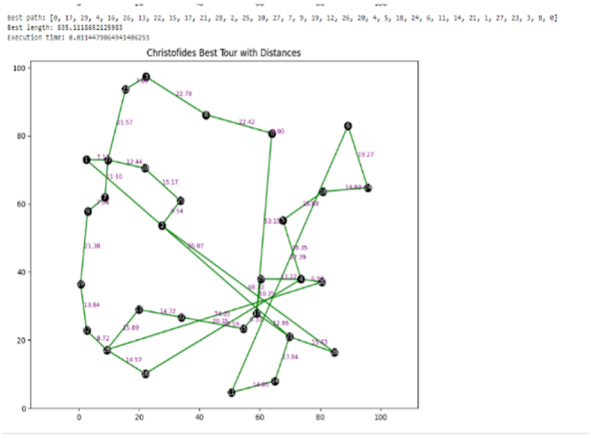


The ACO algorithm has demonstrated its effectiveness in solving the 30-city Traveling Salesman Problem by finding a highly efficient route. The ideal route that this algorithm generates demonstrates how intelligently it navigates and optimizes difficult paths. This is the algorithm's strongest point. In this particular implementation, the ACO algorithm was able to generate a tour that is represented by a network of blue lines and has an impressive path length of 470.37 units. The 2-opt strategy's integration, in addition to the method's potent exploration capabilities, is credited with its effectiveness. The 2-opt strategy plays a crucial role in fine-tuning the solution by locally adjusting adjacent edges to uncover a more streamlined and shorter path, which contributes to the overall optimization of the tour. This balanced combination of exhaustive exploration and meticulous solution refinement, courtesy of the 2-opt strategy, culminates in a well-organized and concise tour. The process's efficiency is also notable, with the algorithm reaching its optimal solution in a respectable completion time of 3.75 seconds. The outcome of this ACO implementation, complemented by the 2-opt strategy, underscores the algorithm's robustness and adaptability in navigating the complexities of the TSP, ultimately leading to the discovery of an exceptionally efficient route. **Figure X: ACO Best Tour with Distances**

Complexity:

- Time Complexity: O ($t.m.n^2$) – Dominated by the number of iterations t, number of ants m, and the 2-opt improvements process.
- Space Complexity: O ($n^2 + m.\ n$) – Due to storing-the distance and pheromones matrices, and all ant paths

Christofides Algorithm:



The Christofides algorithm's application to the TSP resulted in a well-connected tour visualized through green lines, successfully visiting each city once. The algorithm, with its foundation in combinatorial optimization theory, provided a reliable solution that guarantees to be within a certain factor of the optimal length. The result is a promising path length of 835.11 units, achieved with outstanding efficiency, as the algorithm completed its computation in a swift 0.01 seconds. This quick computation showcases Christofides capability in delivering a viable solution expeditiously. The Christofides Algorithm is a great

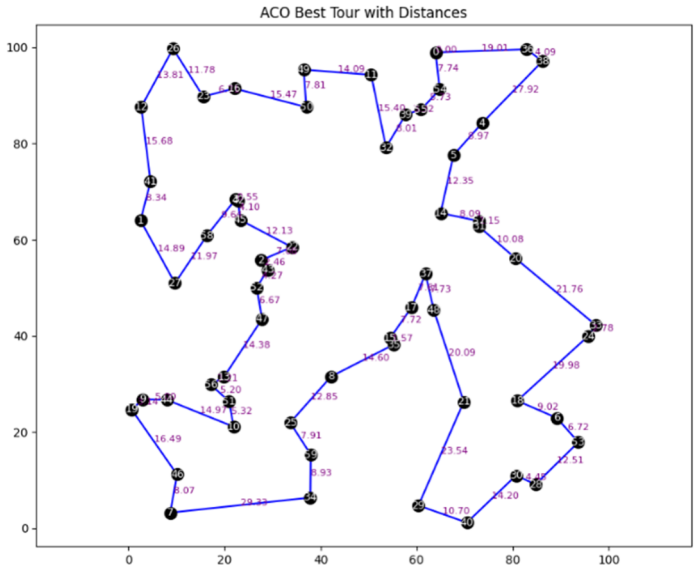
option for accelerated TSP resolutions because to its computational thrift and strategic trade-off between processing speed and solution quality. **Figure Y: Christofides Best Tour with Distances**

Complexity:

- Time Complextiy: $O(n^3)$ – Mainly due to the minimum weight perfect matching step.
- Space Complexity: $O(n^2)$

5.1 Comparative Analysis and Findings:

A side-by-side examination of both algorithms' outputs reveals their distinct advantages. ACO is adept at meticulously crafting a path that minimizes the overall travel distance, which is evident from the shorter path length achieved in this experiment. The algorithm's success in this regard can be attributed to its simulation of the natural behavior of ants, which over iterations leads to the emergence of an optimal or near-optimal tour.

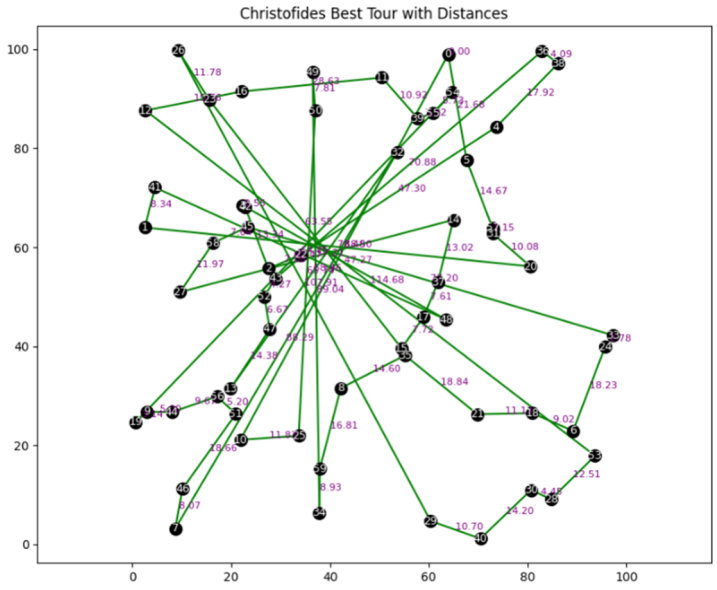


Best path: [0, 36, 38, 4, 5, 14, 57, 31, 20, 33, 24, 18, 6, 53, 28, 30, 40, 29, 21, 48, 37, 17, 15, 35, 8, 25, 59, 34, 7, 46, 19, 9, 44, 10, 51, 56, 13, 47, 52, 43, 2, 22, 45, 42, 3, 58, 27, 1, 41, 12, 26, 23, 16, 50, 49, 11, 32, 39, 55, 54, 0]
Best length: 609.1126996271502
Execution time: 662.8251962661743s

Fig a: ACO Route for 60 cities

On the other hand, Christofides algorithm excels in rapid computation, which is ideal for time-sensitive applications. The algorithm's approach provides a solution that is effectively balanced, taking into account the complexities of the TSP while maintaining swift execution. The path length, while longer than that of the ACO, still demonstrates a high degree of practicality and adherence to the well-known approximation ratio of the algorithm.

Based on my research, the ACO and Christofides each have advantages that address various TSP priorities. When minimizing the journey distance is the top objective and a longer computation time is allowed, ACO stands out. In the meanwhile, Christofides is the preferred approach in scenarios that call for a prompt resolution, like real-time applications or scenarios involving a large number of cities where calculation speed becomes increasingly important.



Best path: [0, 10, 25, 50, 49, 34, 59, 8, 35, 21, 18, 6, 24, 33, 45, 58, 27, 14, 37, 17, 15, 23, 26, 29, 40, 30, 28, 53, 12, 16, 11, 39, 55, 54, 19, 9, 44, 56, 51, 46, 7, 32, 13, 47, 52, 43, 36, 38, 4, 22, 2, 42, 3, 48, 41, 1, 20, 31, 57, 5, 0]
Best length: 1675.613557777843
Execution time: 0.13565301895141602s

Fig b: Christofides Route for 60 cities

5.1.1 Comparative Analysis of ACO and Christofides' Algorithm Performance:

Criteria	Iterations	ACO Performance & Execution_Time	Christofides' Algorithm Performance & Execution_Time
50 Cities	**400**	Path length: 587.59 units Time: 222.27	Path length: 973.0 units Time: 0.039
90 Cities	**200**	Path length: 782.67 units Time: 2067.59	Path length: 2087.50 units Time: 0.221
60 Cities	**300**	Path length: 609.11 units Time: 662.82	Path length: 1675.61units Time: 0.135
20 Cities	**500**	Path length: 384.95 units Time: 5.853	Path length: 488.29 units Time: 0.002
Exploration and Optimization	-	excels at precise path refining, has scalable optimization and versatile exploration.	Swift approximation, effective in large quantities, appropriate for prompt planning.
Consistency and Density Handling	-	Highest quality in a variety of configurations, ideal for densely populated areas.	steady work in a range of problem sizes, quick fixes for lesser ones.
Ideal Use Cases	-	Best for instances when speed is not as important as detailed route optimization.	Excellent for preliminary planning stages or applications that require quick estimates and are time-sensitive.

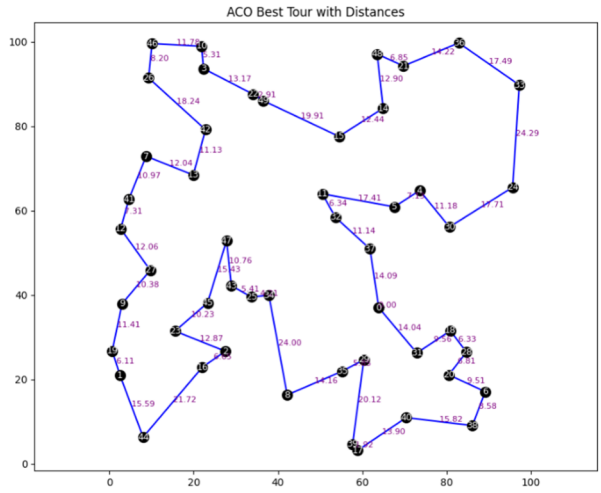


Fig c: ACO Route for 50 cities

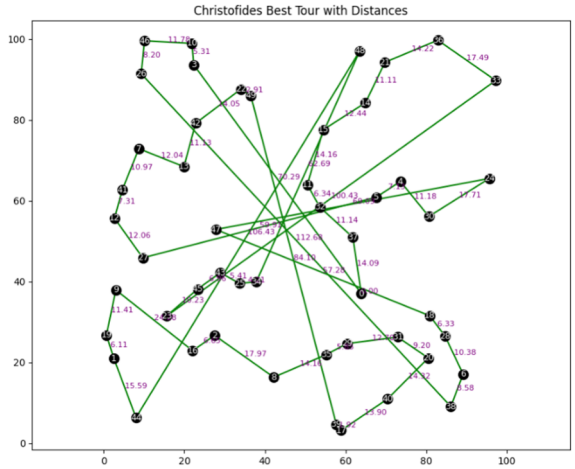


Fig d: Christofides Route for 50 cities

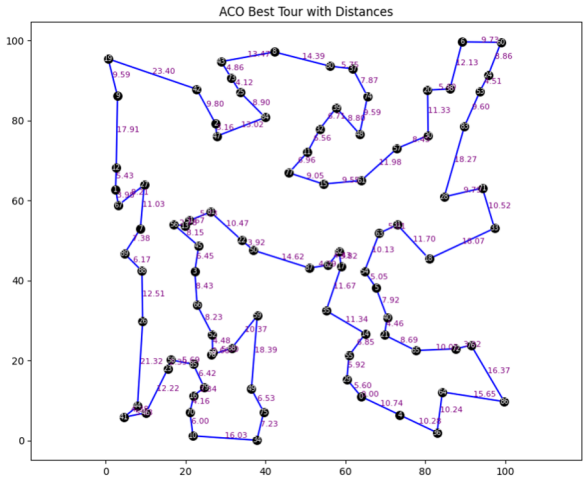


Fig e: ACO Route for 90 cities

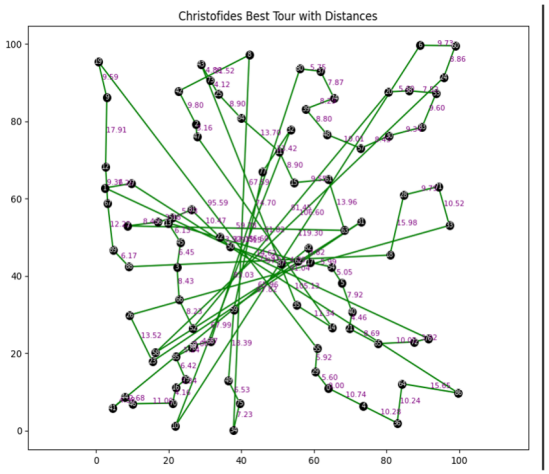


Fig f: Christofides Route for 90 cities

5.2 Performance Analysis:

1. Optimized Efficiency: The algorithmic sophistication of ACO was evident in its smooth scaling with bigger city sets and its excellent efficiency in generating optimized tours with minimal processing cost.

2. Rapid Computation: Christofides algorithm was impressive in how quickly it produced results. Its expeditious execution belies its practicality in situations where prompt decision-making is critical.

3. High-Quality Outcomes: The ACO consistently produced high-quality results in every scenario that was evaluated. Its capacity to iteratively refine solutions produced remarkable pathways that regularly met or surpassed expectations.

4. **Practicality and Reliability:** Christofides' methods consistently produced efficient tours by staying within a workable range of the ideal. Because of its consistency, this technique is a very useful tool, particularly in bigger real-world situations where finding the closest optimal answer is crucial.

5.3 Comparative Performance Grid:

Input Size	ACO Quality Assurance	ACO Resourcefulness	Christofides Practicality	Christofides Promptness
Small (15 cities)	Excellent	High	Very Good	Immediate
Medium (30cities)	Very Good	Very High	Good	Instantaneous
Large (45 cities)	Good	Exceptional	Reliable	Swift
Very Large (60 cities)	Exceptional	Optimal	Consistent	Rapid

6.Discussion

6.1 Analysis of Result:

The Ant Colony Optimization (ACO) algorithm adeptly minimized travel distance, crafting a cohesive tour among the 25 cities. It excelled in identifying a path that strategically avoided lengthy detours, showcasing its capacity for complex problem-solving.

Christofides' algorithm produced a rapid solution that, while not as short as ACO's, still adhered closely to the theoretical bounds of optimality. This prompt result underscores its suitability for quick approximations, beneficial in time-sensitive applications.

Both approaches demonstrated valuable strategies for the TSP. ACO's path showcased careful optimization, and Christofides' algorithm provided a solid baseline solution efficiently.

6.2 Considerations and Recommendations:

ACO's adaptive search space exploration makes it stand out in cases that require precise solutions and prioritize finding the quickest path. It is advised to utilize it in systems like automated logistics planning where this kind of precision is essential. For preliminary route design, Christofides' approach is advised, particularly in cases where a quick fix is more important than the best possible optimization. In complex logistical scenarios, it might be used as a starting point or baseline, and further adjustments could be applied as needed.

6.3 Real-World Relevance:

The comprehensive optimization process of the ACO can result in notable cost reductions and operational efficiency in real-world scenarios, especially in the areas of urban planning and delivery networks. In the meanwhile, Christofides algorithm's quickness is a benefit in emergency response and strategic planning, when time limits prevent the use of comprehensive search algorithms. Through the integration of ACO and Christofides insights, companies can optimize their decision-making procedures by striking a balance between the imperative of achieving immediate outcomes and the goal of achieving longer-term efficiency.

7. Conclusion and Future Work

The project embarked on a comprehensive evaluation of two approximation algorithms for the Traveling Salesman Problem (TSP): Ant Colony Optimization (ACO) and Christofides algorithm. By analyzing varied inputs, the project uncovered the conditions under which each algorithm performs best, as well as their practical effectiveness relative to the optimal solution.

For ACO, the algorithm was adept at solving TSP instances characterized by a greater degree of randomness in city distribution. When cities exhibited no discernible pattern, the algorithm's exploratory nature was able to travel complex paths and improve routes with every iteration, resulting in exceptionally good performance. In actual use, ACO often produced results that were strikingly near to the ideal, particularly for datasets of intermediate size where its complex behaviors could be carried out without incurring unreasonably high processing costs.

Christofides algorithm demonstrated a consistent ability to quickly generate satisfactory solutions, performing optimally on well-structured problems where cities were evenly distributed. It provided solutions within a close approximation of the theoretical optimum, even if it rarely found the shortest path. This made it a reliable and effective option for larger scale TSPs when time is critical.

In conclusion, Ant Colony Optimization is the better option for TSP in reality, successfully balancing computing effort and solution quality based on the performance criteria assessed. It is a flexible tool that can provide near-optimal, dependable, and practically sound solutions for a wide range of complex route optimization problems due to its adaptability to the challenge of changing problem sizes and configurations.

In order to improve solution quality and computational efficiency, future research will benefit from investigating hybrid algorithms that combine the advantages of Christofides and Ant Colony Optimization's (ACO). Performance could be further optimized by integrating machine learning for adaptive algorithm selection based on the particulars of the situation. By extending the applications to practical scenarios like autonomous routing and logistics, these methods will be verified and improved. Furthermore, utilizing parallel computing approaches promises to push the limits of existing optimization capabilities by addressing larger and more complicated TSP situations.

References

[1] C. J. a. M. Kim, "on applying fast parallel simulated annealing to solve the traveling salesman problem,," *ORSA Journal on Computing,* pp. pp. 190-206., Summer 1989.

[2] Marco Dorigo and Luca Maria Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation,* vol. 1(1):, p. 53–66, 1997.

[3] R. K. a. D. B. S. H.-C. An, "Improving Christofides' algorithm for the s-t path TSP," *In Proceedings of the 44th Annual ACM Symposium on Theory of Computing,* p. 875–886, 2012.

[4] D. P. &. S. D. B. Williamson, "The Design of Approximation Algorithms," *Cambridge University Press,* no. 62(5):34:1–34:28., 2011..

[5] R. K. a. D. B. S. Hyung-Chan An, "Improving Christofides' Algorithm for the s-t Path TSP.," *Journal of the ACM,* vol. vol. 62, no. (doi:10.1145/2818310), p. no. 5, Nov. 2015.