Comparison of TSP Solutions on TSPLIB Data

Introduction

The Traveling Salesman Problem (TSP) is a classic optimization problem with applications in logistics, telecommunications, and manufacturing. The goal is to find the shortest possible route that visits all cities once and returns to the starting point. Due to its NP-hard nature, solving TSP becomes computationally expensive as the number of cities increases. This report compares several algorithms on TSP datasets from TSPLIB, focusing on solution quality, runtime, and scalability. The dataset includes vertex counts ranging from 14 to 1000, allowing for a thorough analysis of algorithm performance.

Algorithm Descriptions

We analyzed four algorithms, each suited for different problem scales and constraints:

- **Held-Karp (Dynamic Programming)**: Provides an optimal solution by breaking the problem into smaller subproblems using memoization. However, its time complexity of $O(2^N N^2)$ restricts practical usage to fewer than 20 vertices due to exponential execution time.
- **Greedy Algorithm**: A fast, constructive heuristic that selects the nearest unvisited city. While computationally efficient ($O(N^2)$), its local search nature often leads to suboptimal solutions, especially as problem complexity increases.
- **Simulated Annealing (SA)**: A probabilistic heuristic that explores the solution space more broadly than Greedy, escaping local minima by allowing non-improving moves early on. SA can handle large instances but is sensitive to parameter tuning and provides no solution quality guarantees.
- **Christofides Algorithm**: An approximation algorithm for symmetric TSP with a guaranteed upper bound of 1.5 times the optimal solution, assuming the triangle inequality holds. While efficient ($O(N^3)$), it may underperform on non-metric or asymmetric problems.

Data Segmentation and Metrics

To manage the wide range of vertex counts ($10 \leq N \leq 1000$), we segmented the data into three ranges:

- Small-Scale ($10 \leq N \leq 50$)
- Medium-Scale ($50 < N \leq 200$)
- Large-Scale ($200 < N \leq 1000$)

The key metrics are:

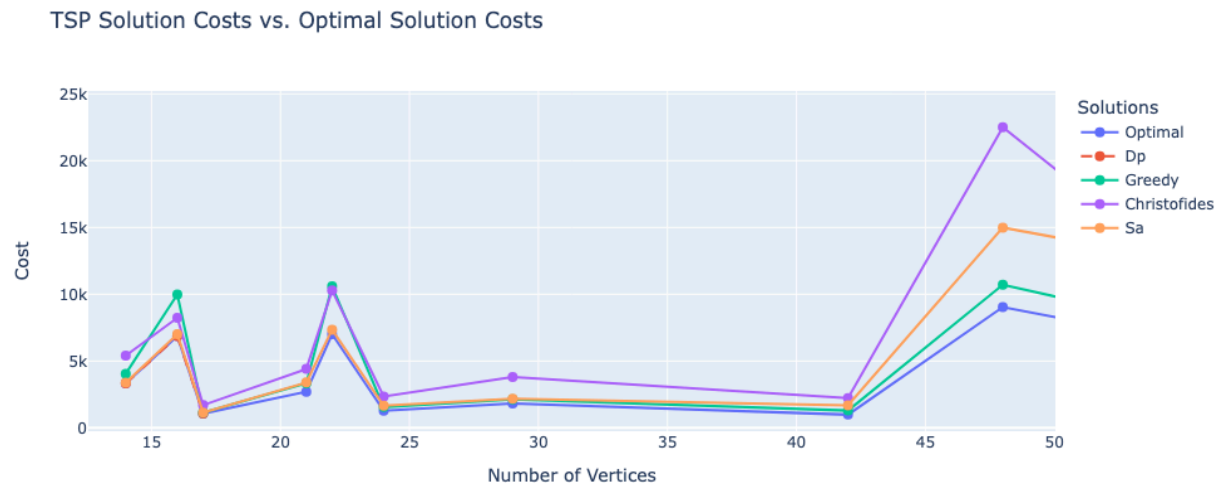
- Solution Cost: Total tour length for each solution.
- Execution Time: Time taken by each algorithm to compute a solution.

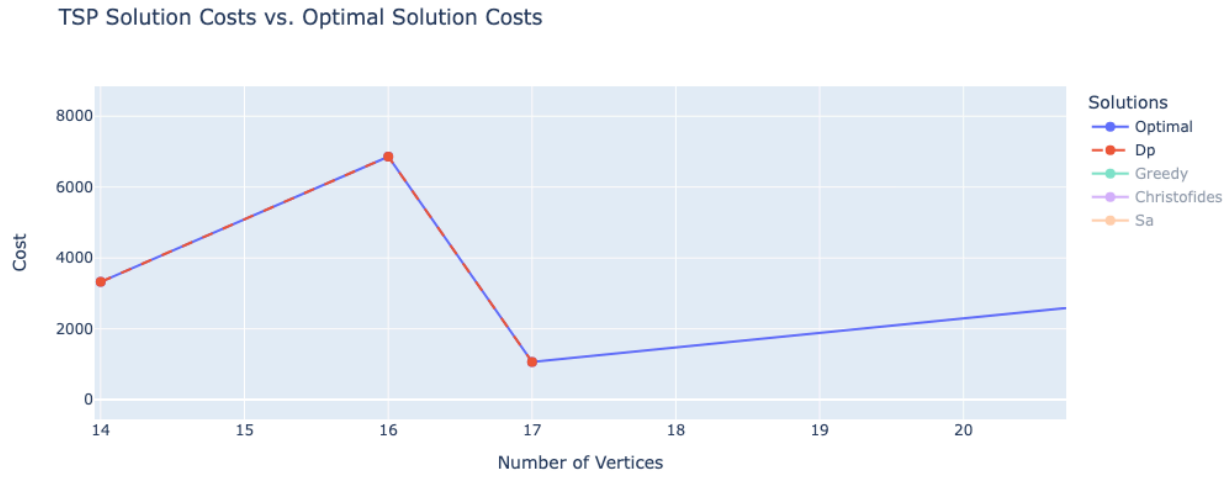
- Deviation from Optimality: Accuracy measured against optimal or best-known values.

Results and Analysis

Solution Cost Trends

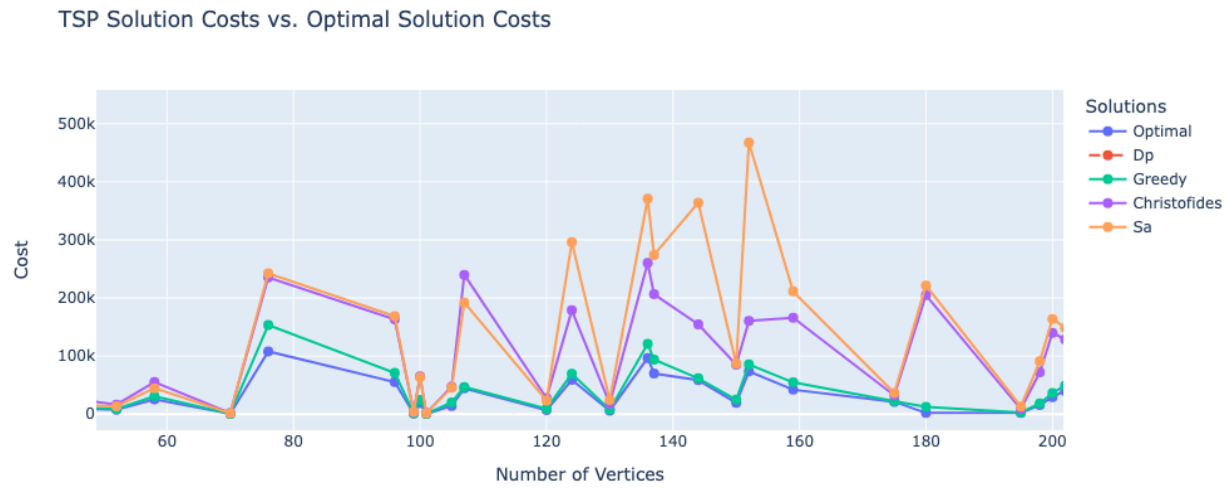
Small-Scale (10 ≤ N ≤ 50):





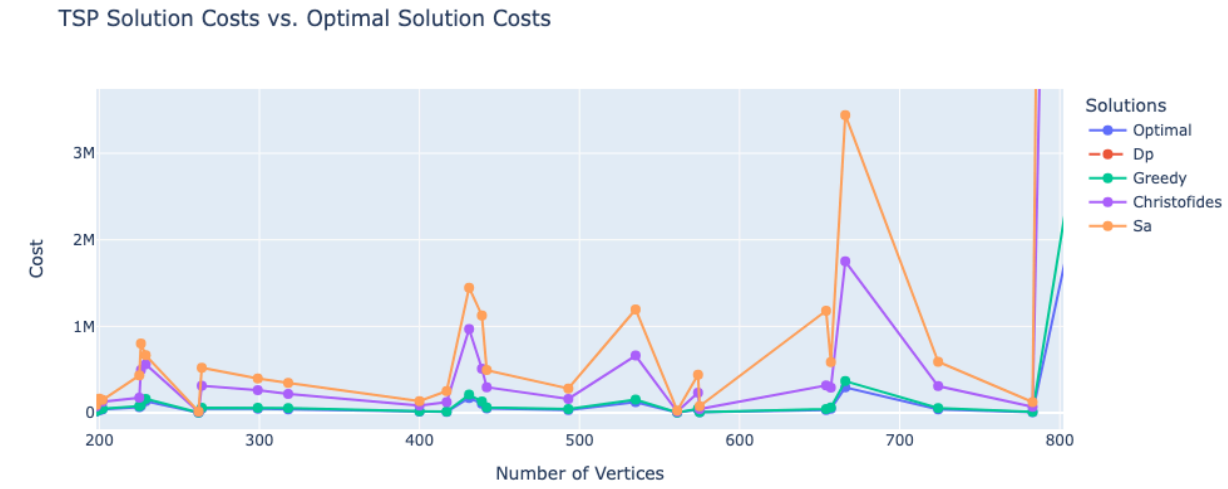
Dynamic Programming (DP) and Greedy performed closely to the optimal solution, with Christofides and SA showing increased deviation as vertex count approached 50.

Medium-Scale (50 < N ≤ 200):



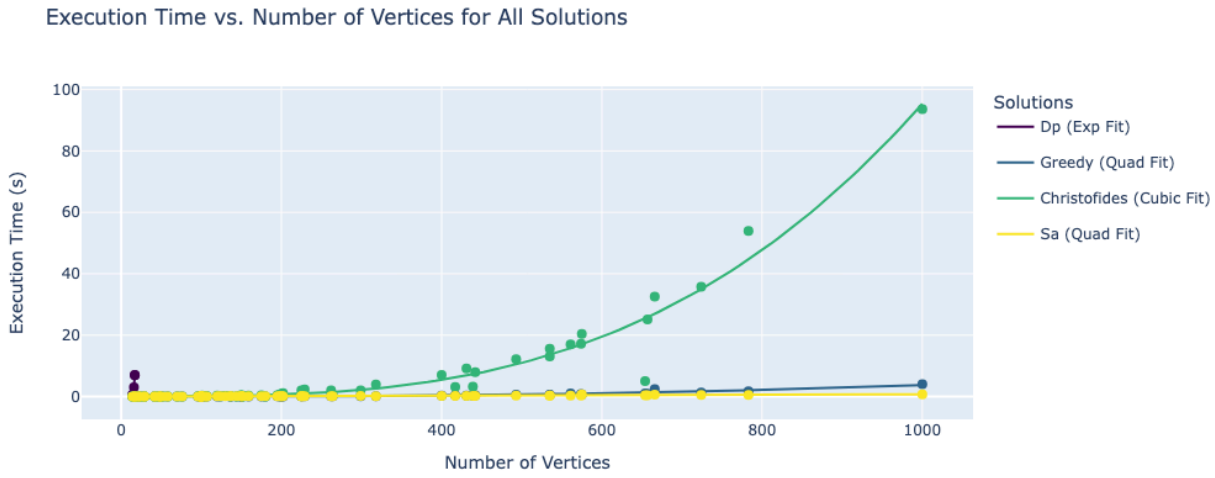
Greedy begins to show significant deviation from the optimal solution as the vertex count increases. Christofides and SA also perform worse, especially beyond 75 vertices, indicating difficulty with larger problem sizes.

Large-Scale (200 < N ≤ 1000):

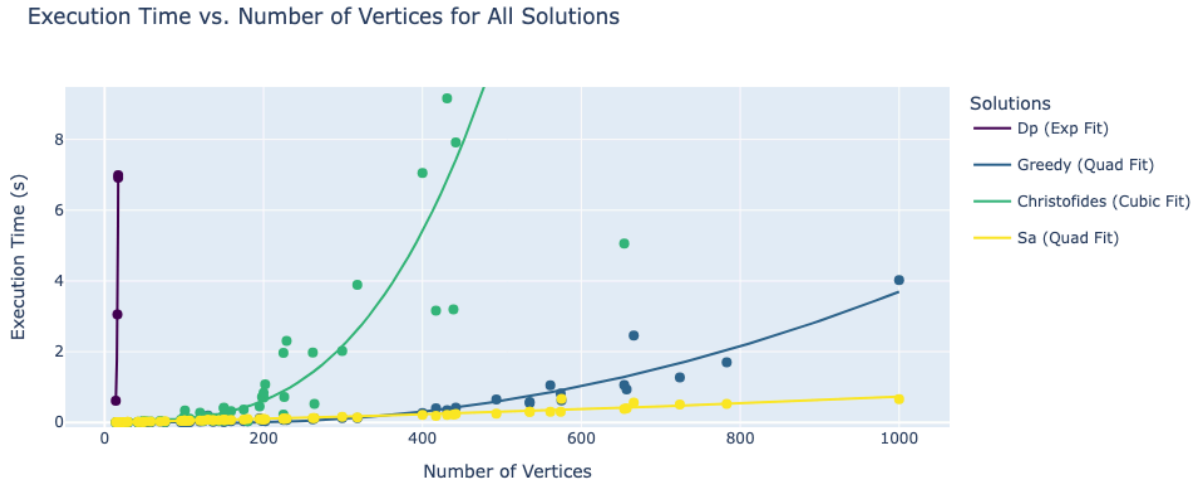


Greedy maintains reasonable proximity to the optimal solution but struggles beyond 700 vertices. Christofides and SA show substantial deviations, with Christofides performing particularly poorly in certain instances (e.g., vertex count around 200 and 700).

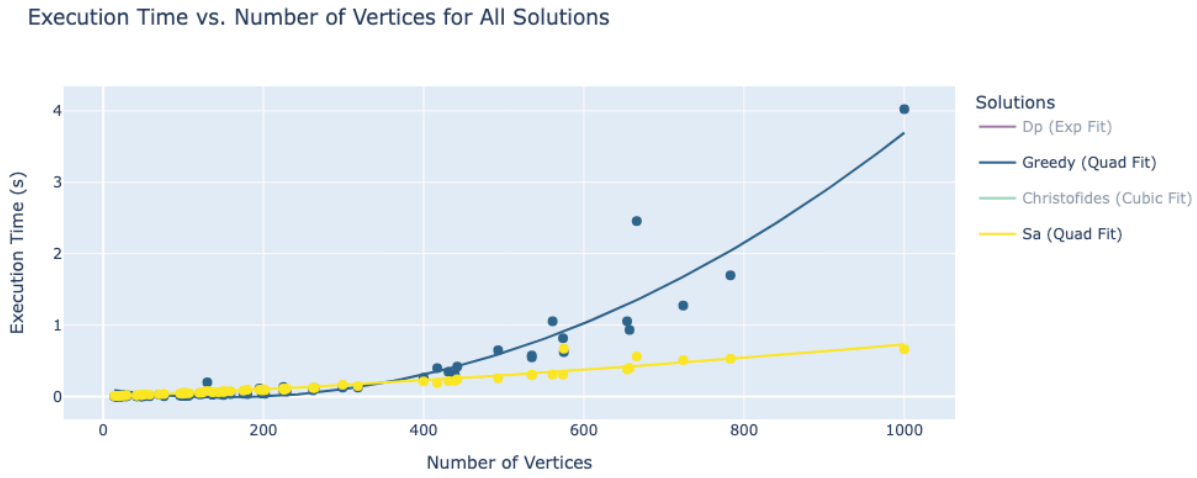
Execution Time and Time Complexity



Christofides: Exhibits cubic growth in runtime, becoming infeasible for graphs larger than 400 vertices.



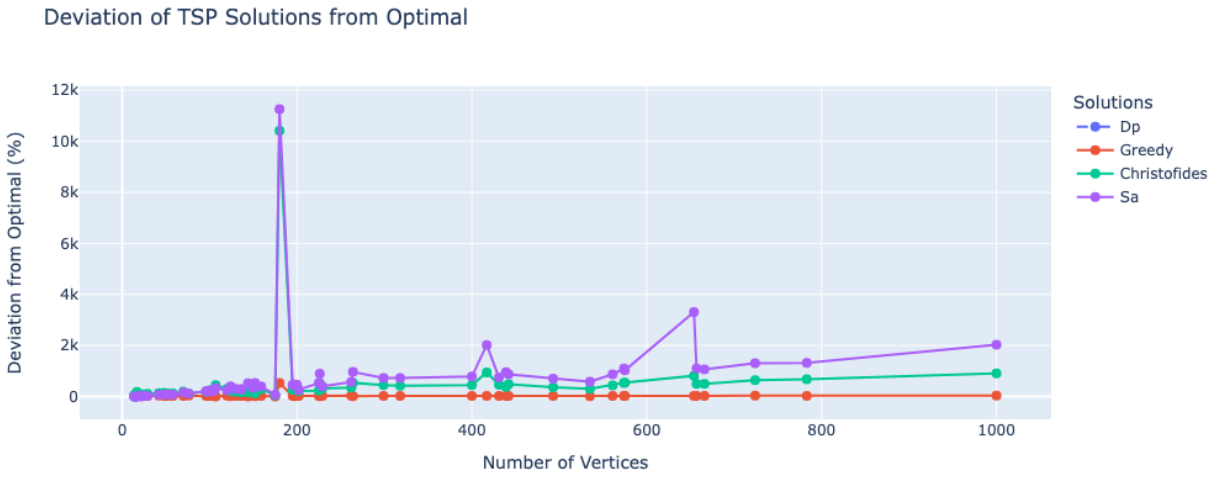
DP: The exponential growth of Held-Karp's execution time renders it impractical beyond 20 vertices.



Greedy and SA: Both maintain quadratic time growth, making them more scalable. SA is slightly slower than Greedy due to its probabilistic nature but offers flexibility in large instances.

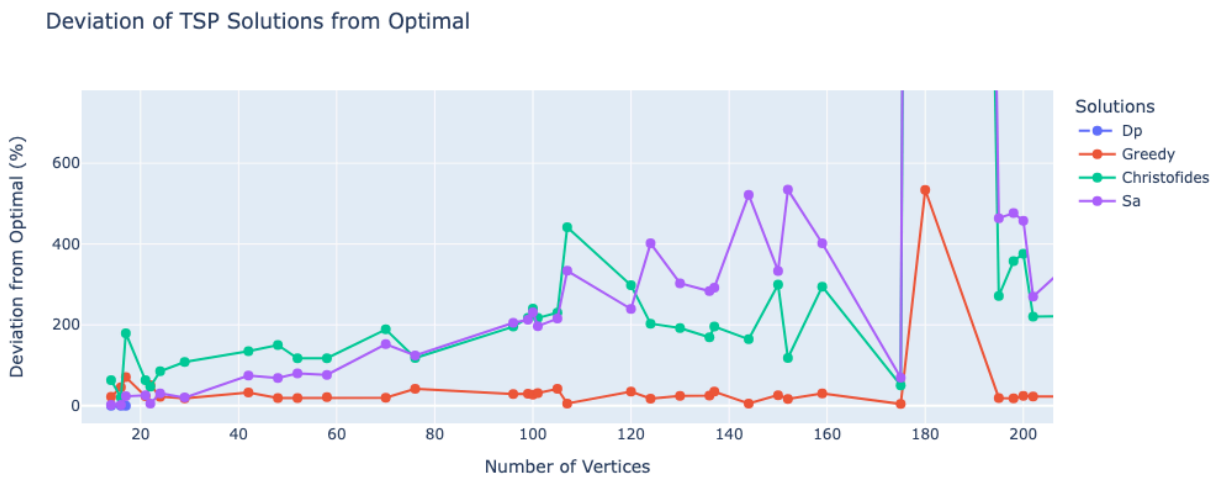
Accuracy and Deviation from Optimality

Large-Scale (200-1000 vertices):



The highest deviations are seen in Christofides, especially at 200 and 700 vertices. SA and Greedy maintain acceptable performance but occasionally exhibit spikes in deviation, particularly around 800 vertices.

Small-Scale (0-200 vertices):



DP provides exact solutions for up to 20 vertices. Greedy remains close to optimal but begins to deviate at larger scales. Christofides and SA show growing deviations, especially past 100 vertices.

Discussion and Insights

Trade-offs Between Accuracy and Speed

The trade-offs are clear: Held-Karp guarantees optimality but is impractical for large instances. Christofides and Greedy offer scalable alternatives but compromise on accuracy. Simulated Annealing provides a flexible balance, adjusting solution quality based on time constraints, though it requires careful parameter tuning.

Scalability

Held-Karp is limited to small problems due to its exponential complexity, while Greedy, Christofides, and SA scale better, offering near-optimal solutions for larger instances. Christofides, despite its theoretical guarantee, struggles on non-metric problems and larger graphs.

Real-World Applicability

Each algorithm has its domain of applicability: Held-Karp for small, exact solutions; Christofides and SA for larger, approximate solutions; and Greedy for settings where speed is a priority, and accuracy can be compromised.

Conclusion

This report provides a comparative analysis of TSP algorithms based on cost, execution time, and accuracy. Christofides and Simulated Annealing offer effective solutions for large-scale instances, balancing computation time with solution quality. Future work could explore parallelization to improve scalability further.