The Travelling Salesperson Problem

Solved by the Simulated Annealing and Tabu Search Algorithms

A Comparative Study of Algorithms
by Siyamthanda Ndlovu

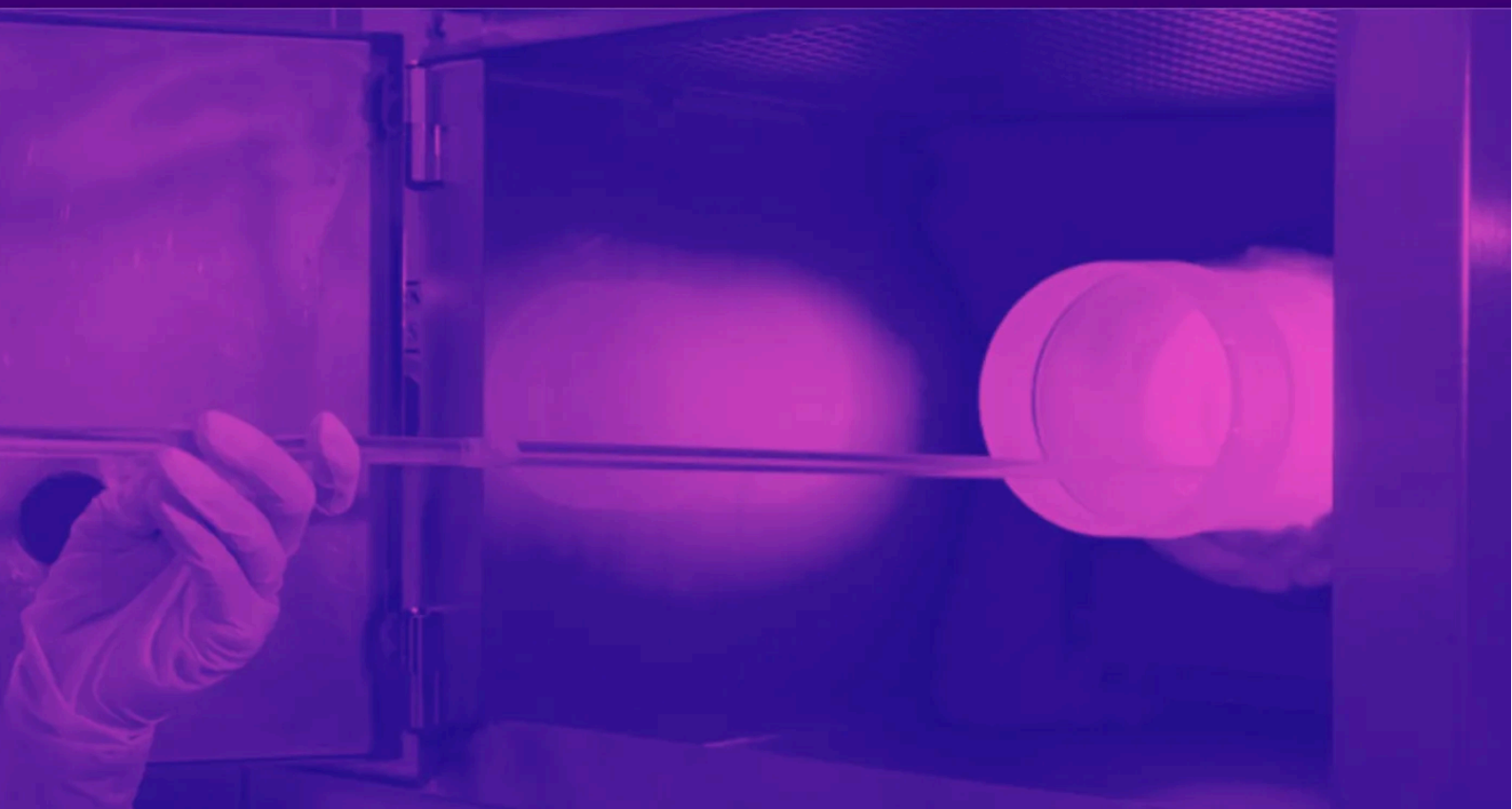

Table of Contents

Introduction

This report evaluates and compares the performance of two local search algorithms, Simulated Annealing (SA) and Tabu Search (TS), in solving instances of the Travelling Salesman Problem (TSP). The study examines key components of each algorithm, including initial solution generation, perturbation methods, neighbourhood definitions, acceptance criteria, and stopping conditions.

Data Representation

The nodes are represented as euclidean coordinates and Euclidean distance was used to find the length between two points. An examples of a problem representation is as follows:

```
# 8 Cities
NAME: tsp_instance_8
TYPE: TSP
DIMENSION: 8
EDGE_WEIGHT_TYPE: EUC_2D
NODE_COORD_SECTION
1 10 90
2 80 10
3 25 50
4 60 75
5 40 20
6 95 65
7 15 35
8 70 85
EOF
```

To ensure a fair comparison, each algorithm was tested under a consistent experimental setup, with relevant parameters defined. Performance was assessed using a structured results table, followed by a critical analysis comparing the two approaches.

Tabu Search

Initial solution generation method	Random solution
Perturbation method	Greedy approach, where only improving moves are accepted
Neighbourhood definition	Randomise a specified percentage of current solution
Stopping criteria	Maximum number of iterations reached

Table 1 : Tabu Search Implementation

Experimental Setup and Algorithm-Specific Parameters

- **Tabu Size :** Set to half the number of cities and remained constant throughout all runs.
- **Neighbour Generation :** Neighbours were created by randomising a percentage of the current solution. Several runs were conducted with the randomisation percentage set to {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}.
- **Seed (%) :** Represents the percentage of the current solution that is changed when generating a neighbour.

Captured Results

This section presents the top 10 results for each problem instance, along with:

- A graph comparing the cost of the top 10 solutions to the time taken to achieve them.
- The average time, cost, and seed for the top 10 runs across different numbers of cities.
- An analysis of the best-performing seed values, highlighting trends and their impact on solution quality.

Tabu Results for 8 Cities

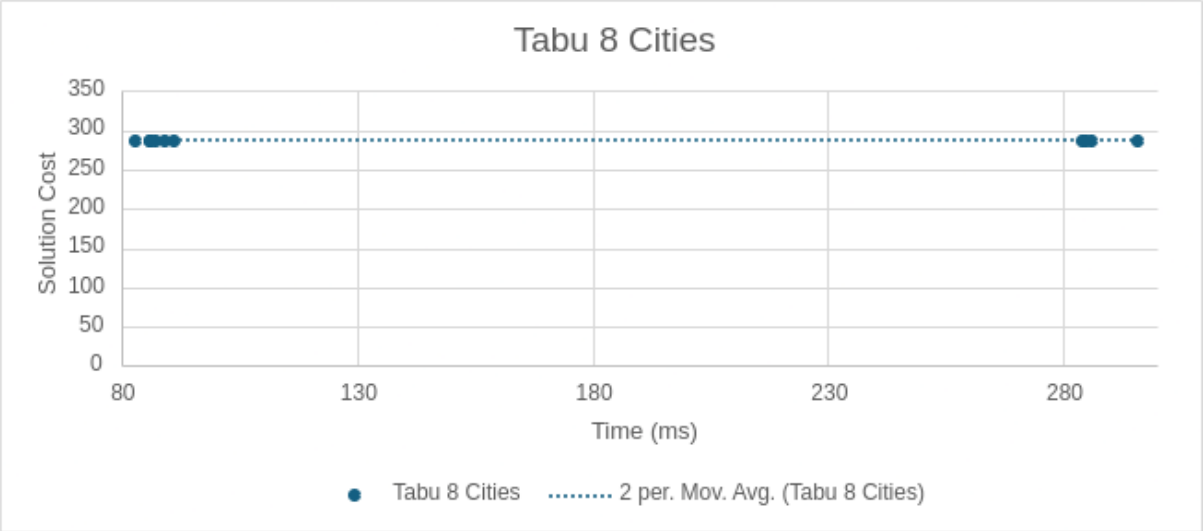

Chart 1 : Average Tabu Results for 8 Cities

Average time (ms)	Average Solution Cost	Average Seed (%)
167.3	286.5016514	80

Table 2 : Average Tabu Results for 8 Cities

Analysis

For 8-city instances, the Tabu Search algorithm consistently produced identical results with an average solution cost of 286.5016514 and an 80% seed similarity. This suggests that TS converges to a specific local optimum, potentially due to strict memory constraints preventing further exploration.

Tabu Results for 12 Cities

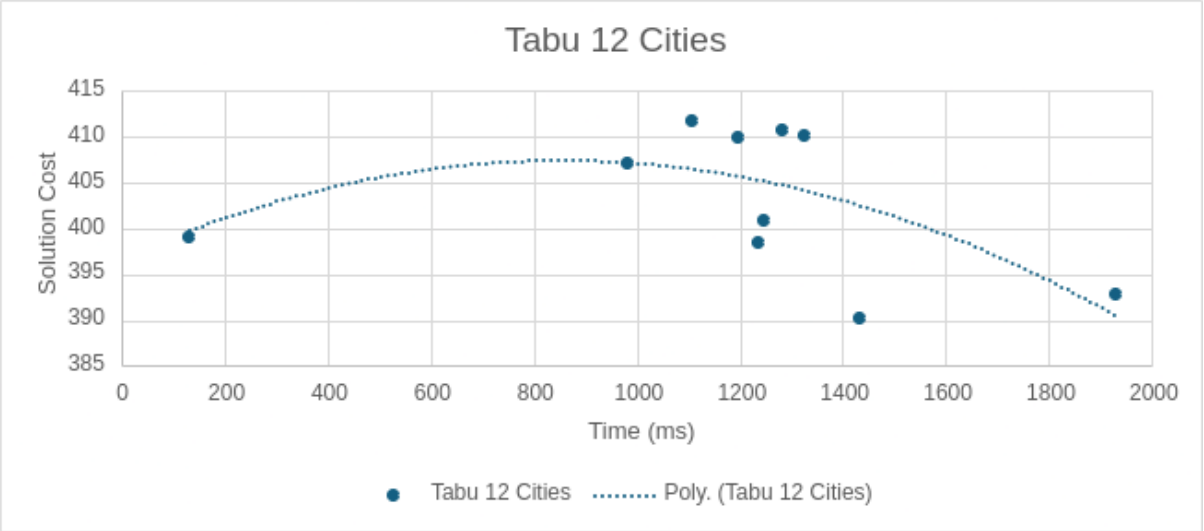

Chart 2 : Average Tabu Results for 12 Cities

Average time (ms)	Average Solution Cost	Average Seed (%)
1186.3	403.1412401	84

Table 3 : Average Tabu Results for 12 Cities

Analysis

For 12-city instances, the best results were found in the 1000–1400 ms range, with some variation. The average solution cost was 403.1412401, clearly indicating that the problem was harder to solve than the smaller case of 8 cities. In most cases, a higher seed percentage (84%) led to better results, similar to what was seen before (see Appendix B). However, there was one outlier where this pattern did not hold, showing that while starting with a good solution usually helps, the randomness of the algorithm can sometimes lead to unexpected results.

Tabu Results for 15 Cities

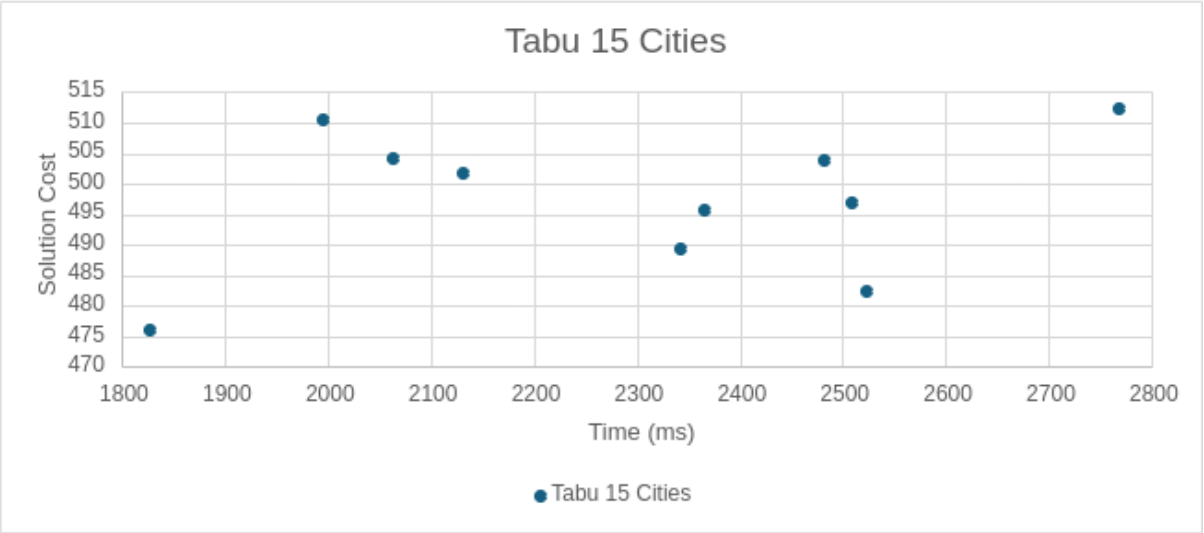

Chart 3 : Average Tabu Results for 15 Cities

Average time (ms)	Average Solution Cost	Average Seed (%)
2301	497.2427333	86

Table 4 : Average Tabu Results for 15 Cities

Analysis

Notably, the average time doubles from 12 cities to 15 cities, showing that the algorithm takes significantly longer as the problem size grows. This is also the third time the average seed value has increased, suggesting that as the number of cities increases, the seed percentage tends to rise as well. However, the consistency ends there, as the relationship between seed values and solution quality is less predictable at this scale. While a higher seed percentage generally aligns with better results, the growing complexity of the problem may introduce more variation in outcomes.

Tabu Results for 20 Cities

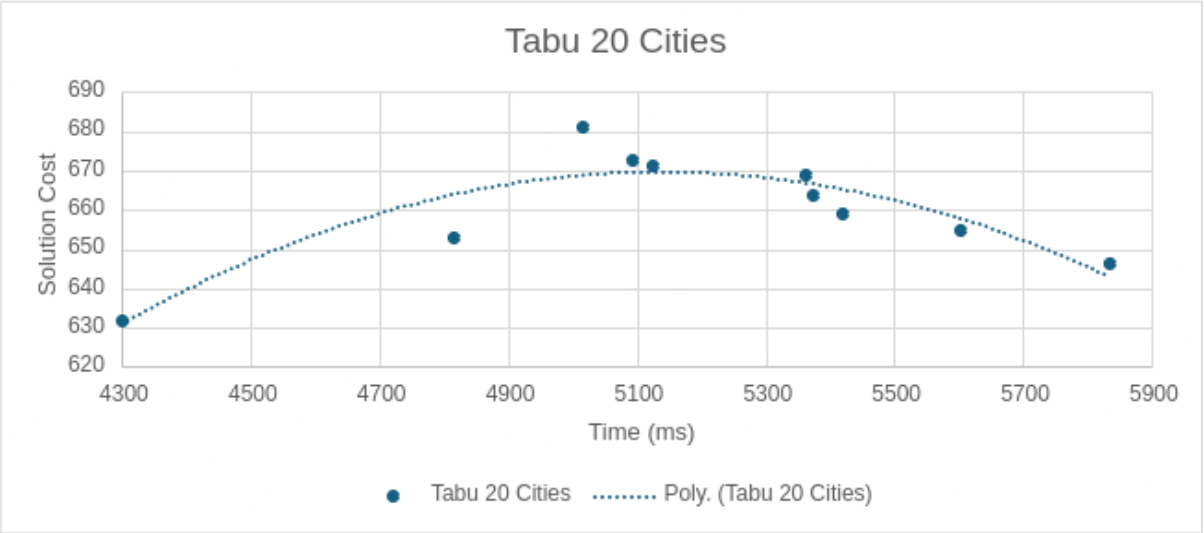

Chart 4 : Average Tabu Results for 20 Cities

Average time (ms)	Average Solution Cost	Average Seed (%)
5194.6	660.2285841	93

Table 5 : Average Tabu Results for 20 Cities

Analysis

The average seed value continues to increase, reaching 93%, while the average time more than doubles again, suggesting an exponential growth in runtime as the problem size increases. Interestingly, the top 10 solutions follow a parabolic trend, indicating a predictable pattern in solution quality. Unlike previous cases, there are no outliers, meaning the algorithm produces consistent results for 20-city instances. This suggests that while the computation time grows rapidly, the search process remains stable and reliable at this scale.

Tabu Results for 25 Cities

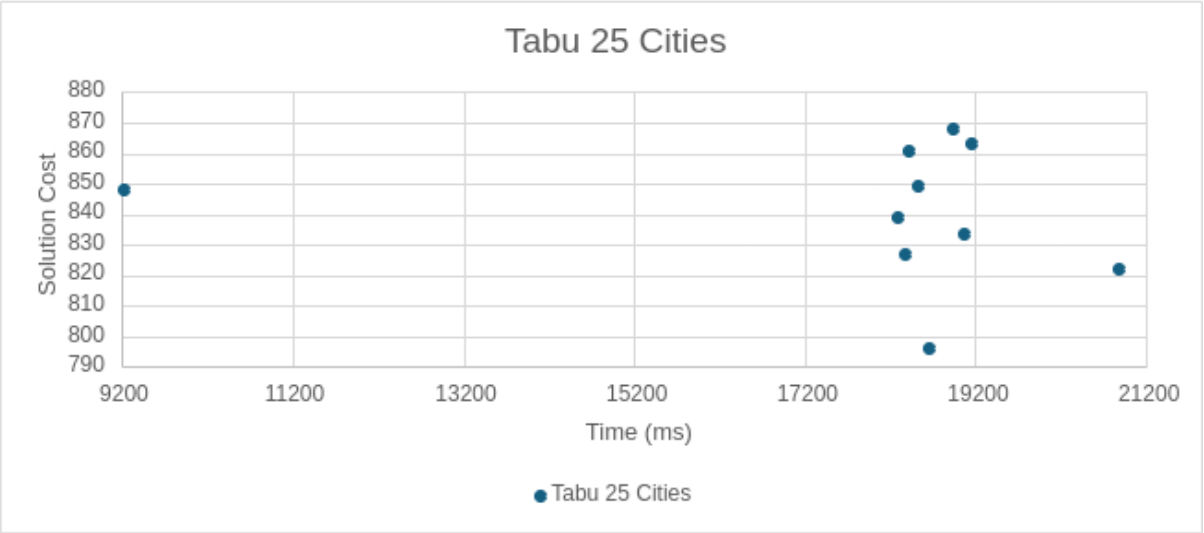

Chart 5 : Average Tabu Results for 25 Cities

Average time (ms)	Average Solution Cost	Average Seed (%)
17956.4	840.4444001	89

Table 6 : Average Tabu Results for 25 Cities

Analysis

Surprisingly, the average seed value drops to 89%, breaking the previous trend of steady increases. However, there is a tight cluster of eight top solutions, suggesting the algorithm frequently converges on similar high-quality results. An outlier appears, with a significantly higher runtime of 9235 seconds, but its seed value closely matches those of other runs (see Appendix B). This suggests that the seed was not the cause of the outlier's extended runtime, implying other factors—such as the algorithm's internal search dynamics or specific problem instance characteristics—played a larger role in its performance.

Best Performing Seed Analysis

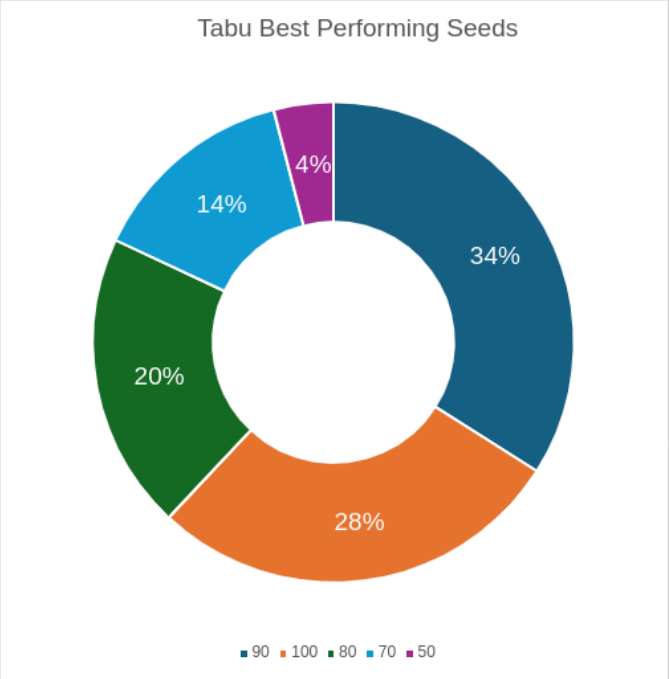

Chart 6 : Pie Chart Indicating the Occurrences of the Best Seed in the Tabu Search Results

For the Tabu algorithm, the best-performing seeds were 90% and 100%, showing that higher seed values led to better solutions. In more than half of the best results, over 90% of the current solution was randomised when generating neighbours. This suggests that a high degree of randomness during the perturbation step helps the algorithm explore the search space more effectively, preventing it from getting stuck in local minima.

Simulated Annealing

Initial solution generation method	Random solution
Perturbation method	Greedy descent approach, only improving moves are accepted, with worse solutions accepted depending on an acceptance criterion
Neighbourhood definition	Randomise a specified percentage of current solution
Acceptance criterion	Accept worse solution as best solution depending on probability influence by temperature
Stopping criteria	Maximum number of iterations reached

Table 7 : Simulated Annealing Implementation

Experimental Setup Algorithm-Specific Parameters

The experiment was conducted using the following setup for the Simulated Annealing (SA) algorithm:

- **Temperature :** The initial temperatures used were (100, 1000, 5000, 10000, 20000).
- **Neighbour Generation :** Neighbours were created by randomising a percentage of the current solution. Several runs were conducted with the randomisation percentage set to {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}.
- **Cooling Cycle :** The *acceptWorseSolution()* method implements a probabilistic approach to determine whether to accept a worse solution in optimization algorithms. It generates a random number and compares it to a threshold based on the current temperature; if the random number is lower than the calculated threshold, it returns true, allowing the acceptance of a worse solution to facilitate exploration of the solution space.
- **Seed (%) :** Represents the percentage of the current solution that is changed when generating a neighbour.

Simulated Annealing Results

For the Simulated Annealing algorithm, the top 10 solutions consistently had identical costs across all problem sizes, indicating that the algorithm quickly converged to a particular solution, regardless of the number of cities involved. This suggests that the annealing process may have found a stable local optimum early in the search, with little variation in the solution quality across runs. Since the results converged, Table 8 covers the average results for 8, 12, 15,20 and 25 cities solved.

Number of cities	Average time (ms)	Average solution cost	Average seed (%)	Average temperature
8	4.4	286.5016514	90	5230
12	14	401.6919473	100	100
15	14	502.7187932	100	20000
20	22	614.0940825	60	1000
25	43	807.9037467	90	10000

Table 8 : Average Simulated Annealing Results

Since the results converged, Table 8 presents the average results for the problem instances involving 8, 12, 15, 20, and 25 cities. These averages represent the typical performance across multiple runs, reflecting the consistency of the Simulated Annealing algorithm in producing stable solutions, despite varying problem sizes. The table includes key metrics such as average time, average solution cost, and average seed percentage, providing a comprehensive view of how the algorithm performs across different city counts.

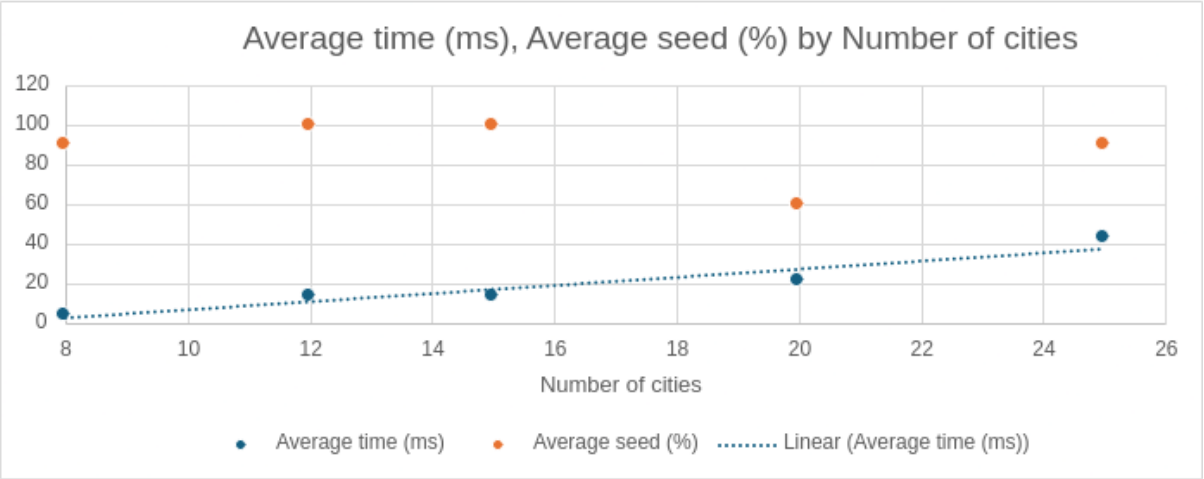

Chart 7 : Average time and seed by number of cities

For Simulated Annealing, the time appears to increase seemingly linearly as the number of cities increases, indicating that the algorithm's runtime grows steadily with problem size. However, the average seed percentage does not show any clear correlation with the number of cities. The variation in seed values seems inconsistent, suggesting that the annealing process is not strictly dependent on the number of cities, and other factors may be influencing the randomisation of the solution. This lack of correlation between seed and city count highlights the stochastic nature of the algorithm, where the randomness in the search process may lead to unpredictable seed values across different runs. More information on the seed usage can be found in Appendix C.

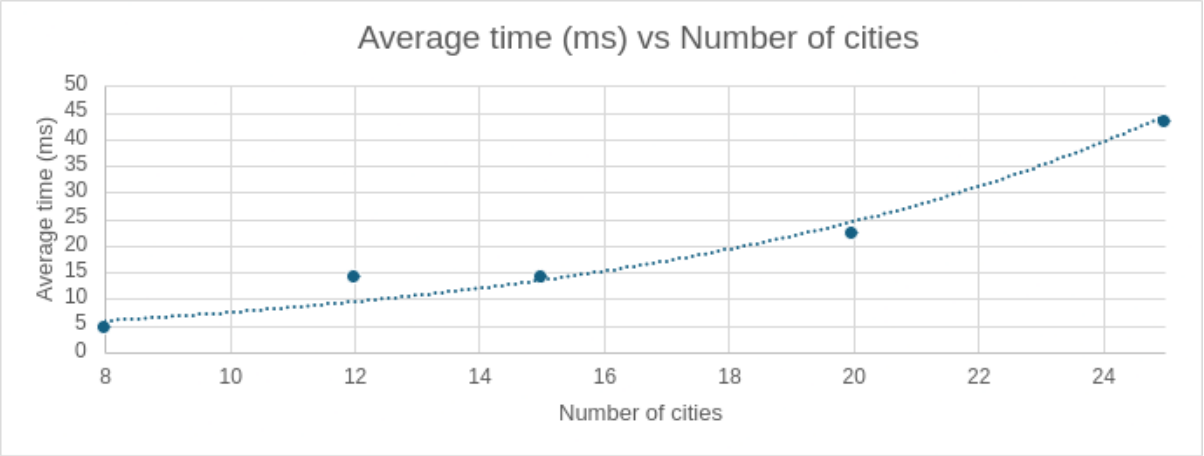

Chart 8 : Average Time vs Number of Cities

The average time to find the best solution increases as the number of cities are increasing for Simulated Annealing, which aligns with expectations. The increase in time appears to be either linear or slightly exponential, suggesting that as the problem size grows, the algorithm requires progressively more time to process larger instances. This trend is typical for local search algorithms like annealing, where the complexity of the search space increases with the number of cities, leading to longer runtimes as the solution space expands.

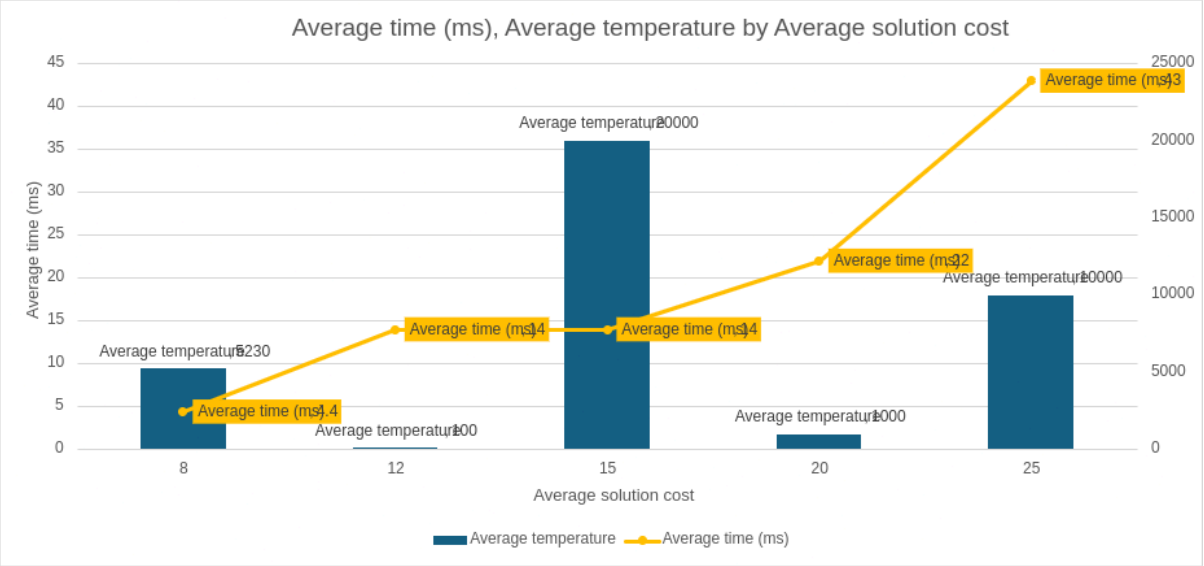

Chart 9: Average time and temperature vs average solution cost

While the average time consistently increases as the number of cities increases there is no clear correlation between temperature and performance. The best-performing temperature fluctuates, suggesting that higher temperatures do not always lead to better results. This irregular pattern makes it unclear whether the behaviour is purely random or if it follows an underlying trend. Additional testing would be needed to determine whether this variation is due to stochastic effects or if certain temperature values are genuinely more effective in different scenarios.

Best Performing Seed Analysis

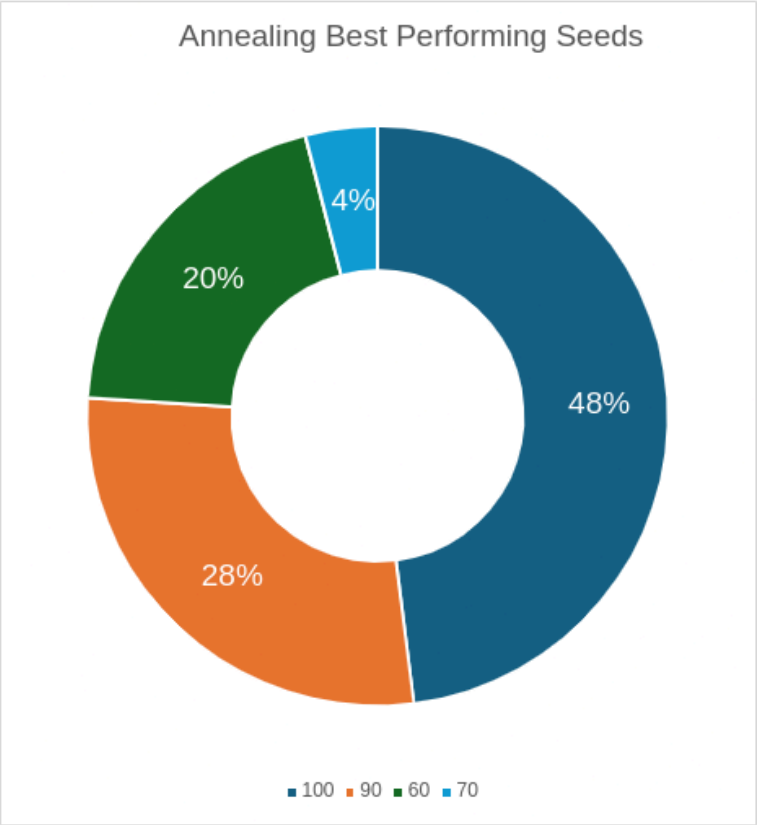

Chart 10 : Pie Chart Indicating the Occurrences of the Best Seed in the Simulated Annealing Results

Unlike in Tabu Search, only four different seeds appeared in the best Simulated Annealing results. Notably, nearly half (48%) of the best-performing solutions (see Chart 10) used a 100% seed, meaning the entire solution was randomised when generating a neighbour. Another 28% of the best results used a 90% seed, which also introduces significant randomness into the search. This suggests that high randomness in neighbour generation plays a crucial role in finding optimal solutions for Simulated Annealing. More details on seed usage can be found in Appendix C.

Algorithm Result Comparison

Best Solutions of Annealing and Tabu

Problem Instance	Algorithm	Seed (%)	Cost	Best Solution	Runtime (ms)
8 Cities	Tabu	80	286.5016514	1-3-7-5-2-6-8-4	83
	Annealing	70	286.5016514	1-3-7-5-2-6-8-4	4
12 Cities	Tabu	80	390.2456492	1-5-3-7-9-4-10-8-2-12-6-11	1432
	Annealing	100	401.6919473	1-5-11-8-12-6-2-4-10-9-7-3	14
15 Cities	Tabu	100	476.0060909	1-14-11-4-13-5-10-12-15-6-8-2-9-3-7	1827
	Annealing	100	502.7187932	1-7-13-15-2-8-9-14-4-11-6-5-12-10-3	14
20 Cities	Tabu	100	631.7361072	1-18-9-20-5-8-12-10-15-4-11-13-2-17-7-14-3-6-16-19	4300
	Annealing	60	614.0940825	1-9-3-6-18-19-8-2-5-20-11-13-15-4-10-17-7-14-12-16	22
25 Cities	Tabu	80	796.0115566	1-5-12-7-3-14-24-21-16-11-15-20-25-6-23-8-10-4-18-13-2-22-19-17-9	18650
	Annealing	90	807.9037467	1-21-24-7-12-9-17-2-19-3-16-11-10-8-4-18-6-13-14-23-20-15-25-22-5	43

Table 9 : An overview of the Problem Instances, Algorithms, Seeds, Best Solution, Best Solution Costs as well as Runtime

When constructing Table 9, if multiple results had the same cost, the result with the shortest runtime was selected to provide clearer insights. This approach highlights the most efficient solutions while maintaining accuracy in performance comparisons. More details on the results can be found in Appendix A.

Average Solutions and Times for Annealing and Tabu

Cities	Annealing Average of cost	Annealing Average of time (ms)	Tabu Average of cost	Tabu Average of time (ms)
8	375.3818027	5.08	379.2508899	95.34
12	522.6145012	9.66	517.0232913	594.04
15	625.8584615	15.08	626.1780673	1394.15
20	808.8034811	25.42	817.9313169	3730.12
25	1024.163092	43	1014.971781	13804.81

Table 10 : Average Annealing and Tabu Results for All Problem Instances

Algorithm Performance Analysis

Table 10 compares the average solution cost and runtime for Simulated Annealing and Tabu Search across different problem sizes. Annealing consistently achieves lower runtimes, often by a significant margin, while solution costs remain similar for both algorithms. Interestingly, Tabu Search occasionally finds slightly better solutions (e.g., at 12 and 25 cities), but its runtime grows exponentially as the number of cities increases.

In contrast, Simulated Annealing scales more gradually, maintaining a steady increase in computation time. At larger problem sizes, the cost difference between the two algorithms narrows, suggesting that Tabu Search may be more effective for larger instances, despite its higher computational cost. This highlights a clear trade-off between solution quality and runtime, where Annealing provides faster, slightly less optimal solutions, while Tabu Search sacrifices speed for better results.

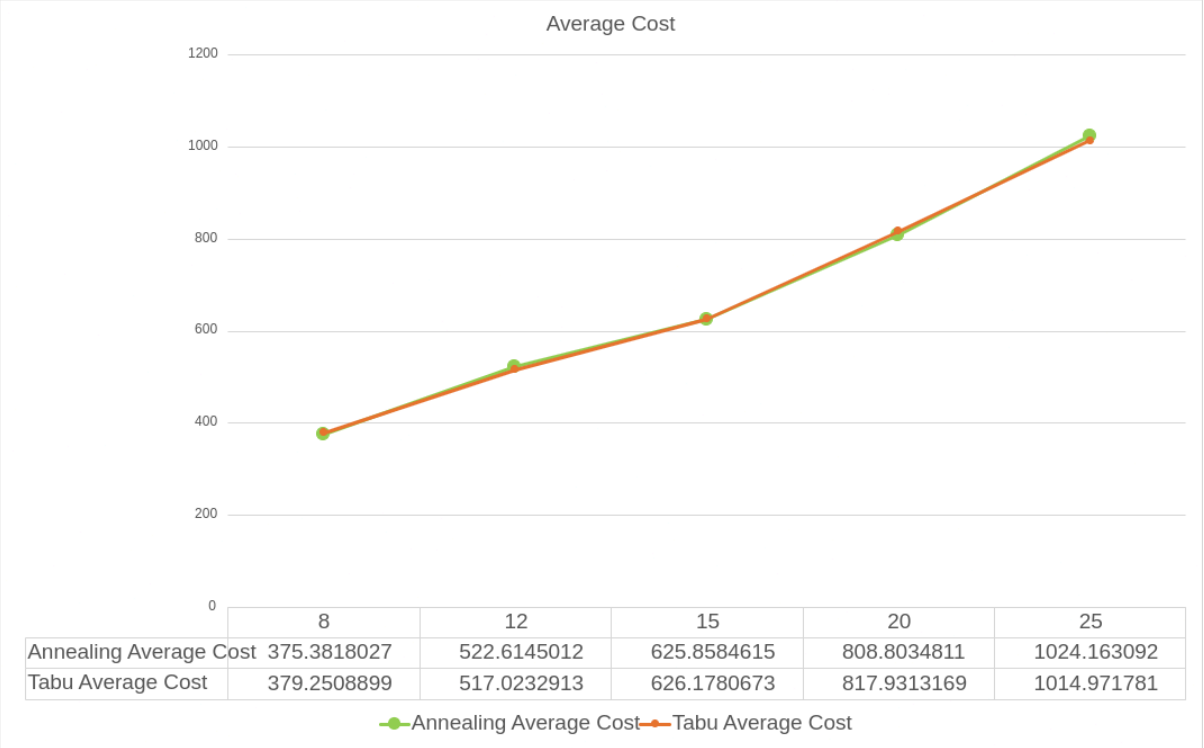

Chart 11 : Average of the best cost found by both algorithms per number of cities

As seen in Table 10 and Chart 11, for larger problem sizes, the cost difference between the two algorithms narrows but Tabu Search is slightly more effective for solving larger instances. However, the improvement in cost achieved by Tabu Search is minimal, indicating that both algorithms are capable of finding near-optimal solutions.

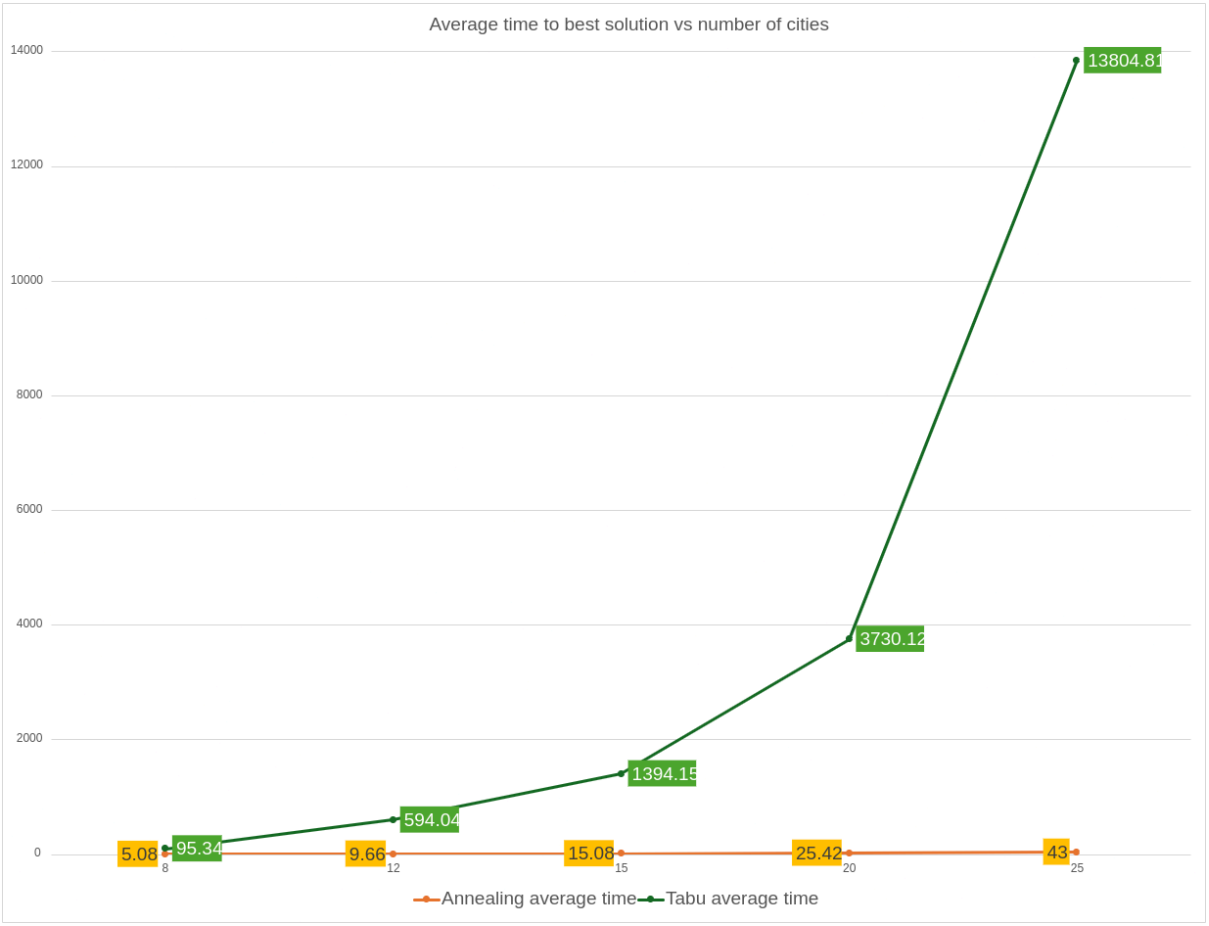

Chart 12 : Average time to find best solution vs number of cities

From Table 10 and Chart 12, it is evident that while Tabu Search achieves slightly lower cost values, its significantly higher runtime may not always justify the additional computational expense. The marginal improvement in solution quality comes at a considerable cost in time, making Simulated Annealing a more practical choice for applications where a balance between speed and solution quality is required.

This suggests that unless absolute optimality is necessary, Simulated Annealing provides a more efficient trade-off between performance and computational resources.

Algorithm Implementation Improvements

Smarter Neighbour and Initial Solution Generation

Instead of selecting a single random initial solution or neighbour, a better approach would be to generate multiple candidates and choose the one with the lowest cost. This would increase runtime for both algorithms but could significantly improve solution quality. Future experiments should track how much additional runtime this method requires and whether the trade-off is worthwhile.

Use Plateau as Stopping Condition

To reduce unnecessary computation time, the algorithm could terminate once a plateau is reached in the best solution. In Simulated Annealing, if a plateau occurs when the temperature is very low, it indicates that worse solutions are unlikely to be accepted, making further iterations redundant. Implementing this condition could help save computational resources without sacrificing solution quality.

Conclusion

This study compared Simulated Annealing and Tabu Search in solving instances of the Travelling Salesman Problem (TSP). The results showed that while Tabu Search occasionally found slightly better solutions, it did so at the cost of significantly longer runtimes, especially as the problem size increased. Simulated Annealing, on the other hand, maintained a much lower runtime while still finding near-optimal solutions, making it the more practical choice for balancing efficiency and solution quality.

Additionally, analysis of the best-performing seeds suggested that higher randomness in neighbour generation (90–100%) tended to produce the best results for both algorithms. Further improvements, such as smarter neighbour selection and plateau-based stopping criteria, could enhance performance and reduce unnecessary computation.

Overall, the findings highlight that while Tabu Search may be useful for high-precision solutions, Simulated Annealing remains a strong alternative for cases where speed and efficiency are more important than slight improvements in cost.

Appendices

Appendix A : Full Results in Excel Sheet

This spreadsheet was used to parse and generate charts for the experimental results. The charts in this report are distributed across multiple sheets, each containing specific comparisons and breakdowns of the data. It includes the following sheets :

1. **Compare Both** – Direct performance comparison of both algorithms, including graphs that aid in understanding the time complexity of the algorithms
2. **Best Solutions Separate** – Individual best solutions, grouped by number of cities
3. **Best Solutions Combined** – Combined list of solutions for Tabu Search and Simulated Annealing
4. **Best Solutions Tabu** – Top results for Tabu Search
5. **Best Solutions Annealing** – Top results for Simulated Annealing
6. **Combined Solutions** – Raw data, excluding columns for text solutions and number of iterations to facilitate filtering and sorting data
7. **Combined Solution Raw** – Unprocessed raw data

The link to the spreadsheet :
TSP x Simulated Annealing & Tabu Search.xlsx

Appendix B : Top 10 Solutions for Tabu Search

This section contains the top 10 results for Tabu Search, grouped by the number of cities.

Time	Cost	Seed	Tabu Size	Solution
83	286.5016514	80	4	1-3-7-5-2-6-8-4
86	286.5016514	70	4	1-4-8-6-2-5-7-3
86	286.5016514	70	4	1-4-8-6-2-5-7-3
87	286.5016514	70	4	1-4-8-6-2-5-7-3
89	286.5016514	70	4	1-4-8-6-2-5-7-3
91	286.5016514	80	4	1-4-8-6-2-5-7-3
284	286.5016514	90	4	1-3-7-5-2-6-8-4
285	286.5016514	90	4	1-4-8-6-2-5-7-3
286	286.5016514	90	4	1-3-7-5-2-6-8-4
296	286.5016514	90	4	1-3-7-5-2-6-8-4
167.3	**286.5016514**	**80**		

Table B-1 : Tabu Solutions for 8 Cities

Time	Cost	Seed	Tabu Size	Solution
1432	390.2456492	80	6	1-5-3-7-9-4-10-8-2-12-6-11
1931	392.8919805	70	6	1-11-8-12-6-2-4-10-7-9-3-5
1237	398.51279	90	6	1-5-3-7-9-8-6-12-2-10-4-11
130	398.9869256	50	6	1-12-6-9-7-4-10-2-8-11-3-5
1247	400.9741426	80	6	1-5-3-7-9-4-10-6-12-11-8-2
980	407.0692254	100	6	1-11-2-8-5-3-7-9-4-10-6-12
1195	409.933082	100	6	1-12-6-2-4-10-9-7-11-8-3-5
1324	410.1758907	100	6	1-5-11-6-12-8-2-10-4-7-9-3
1282	410.7868768	80	6	1-9-7-3-5-11-8-2-10-4-6-12
1105	411.8358378	90	6	1-5-11-12-6-8-4-10-2-9-7-3
1186.3	**403.1412401**	**84**		

Table B-2 : Tabu Solutions for 12 Cities

Time	Cost	Seed	Tabu Size	Solution
1827	476.0060909	100	7	1-14-11-4-13-5-10-12-15-6-8-2-9-3-7
2524	482.301911	80	7	1-10-12-5-8-4-13-6-2-15-11-14-9-3-7
2343	489.3622164	90	7	1-4-11-13-15-8-2-6-5-12-10-3-14-9-7
2366	495.7517253	100	7	1-7-3-13-8-12-10-5-6-2-15-11-4-14-9
2509	496.7828713	90	7	1-14-11-4-13-15-6-2-5-10-7-12-8-9-3
2132	501.7233599	50	7	1-7-9-3-10-5-2-6-15-4-11-14-13-8-12
2482	503.6927909	80	7	1-7-3-12-2-15-6-8-5-10-9-11-4-14-13
2063	504.2070671	90	7	1-10-12-2-8-5-3-7-9-4-13-6-15-11-14
1996	510.4734016	90	7	1-10-12-5-15-6-2-8-9-13-3-11-4-14-7
2768	512.1258981	90	7	1-7-5-9-13-11-15-6-2-8-12-10-3-4-14
2301	**497.2427333**	**86**		

Table B-3 : Tabu Solutions for 15 Cities

Time	Cost	Seed	Tabu Size	Solution
4300	631.7361072	100	10	1-18-9-20-5-8-12-10-15-4-11-13-2-17-7-14-3-6-16-19
5834	646.2397507	90	10	1-9-10-7-2-14-3-19-17-12-8-18-4-15-11-20-5-13-6-16
4815	653.0067509	90	10	1-16-19-6-14-8-5-15-4-18-11-13-10-12-20-17-7-2-3-9
5602	654.8778504	100	10	1-15-6-19-7-17-2-9-16-3-12-14-8-13-5-20-11-10-4-18
5421	658.7638063	90	10	1-9-13-11-18-15-20-17-2-5-7-8-3-14-6-12-19-16-10-4
5375	663.8608589	90	10	1-9-3-12-15-18-4-13-11-16-14-10-2-7-8-17-5-20-19-6
5362	668.6281602	100	10	1-19-9-16-17-8-14-15-4-18-6-3-12-10-2-7-5-13-20-11
5125	671.3873885	100	10	1-14-10-17-2-7-8-13-4-18-20-5-11-15-3-16-19-12-6-9
5095	672.6641831	70	10	1-3-16-6-10-19-2-17-20-13-7-12-14-8-15-11-5-4-18-9
5017	681.1209843	100	10	1-12-6-15-4-20-18-13-11-5-2-10-7-19-17-8-14-3-9-16
5194.6	**660.2285841**	**93**		

Table B-4 : Tabu Solutions for 20 Cities

Time	Cost	Seed	Tabu Size	Solution
18650	796.0115566	80	10	1-5-12-7-3-14-24-21-16-11-15-20-25-6-23-8-10-4-18-13-2-22-19-17-9
20885	821.7047881	70	10	1-24-2-22-5-7-19-9-17-11-8-10-4-14-23-20-13-15-18-6-25-12-16-3-21
18379	826.7099177	90	10	1-24-16-21-12-14-7-15-10-6-18-2-17-22-3-5-9-19-4-11-23-25-13-20-8
19076	833.4478153	80	10	1-16-3-12-5-9-21-24-7-14-10-15-6-4-25-20-18-23-17-19-13-2-22-8-11
18290	838.6436313	80	10	1-24-9-5-3-14-13-25-19-17-4-2-15-6-23-8-20-10-18-22-12-7-16-11-21
9235	847.860603	100	10	1-24-21-19-17-6-18-5-9-16-14-3-12-7-25-2-22-15-20-13-23-8-4-10-11
18527	849.0343331	90	10	1-24-16-4-18-6-23-11-21-14-8-3-7-12-5-20-19-22-25-15-10-17-2-9-13
18416	860.3400062	100	10	1-24-12-2-6-9-7-19-17-22-15-20-5-3-13-18-25-23-10-4-14-11-16-8-21
19155	862.9417169	100	10	1-20-18-15-8-14-11-6-23-10-17-16-4-25-19-9-7-5-2-22-13-3-12-24-21
18951	867.7496331	100	10	1-14-16-21-5-20-10-11-4-8-25-3-12-7-24-13-2-9-19-22-6-18-23-15-17
17956.4	**840.4444001**	**89**		

Table B-5 : Tabu Solutions for 25 Cities

Appendix C : Top 10 Solutions for Simulated Annealing

This section contains the top 10 results for Simulated Annealing, grouped by the number of cities.

Time	Cost	Seed	Temperature	Solution
4	286.5016514	70	100	1-3-7-5-2-6-8-4
5	286.5016514	90	100	1-4-8-6-2-5-7-3
7	286.5016514	100	100	1-4-8-6-2-5-7-3
4	286.5016514	90	1000	1-3-7-5-2-6-8-4
4	286.5016514	100	1000	1-3-7-5-2-6-8-4
4	286.5016514	90	5000	1-3-7-5-2-6-8-4
5	286.5016514	100	5000	1-3-7-5-2-6-8-4
4	286.5016514	70	10000	1-4-8-6-2-5-7-3
3	286.5016514	100	10000	1-4-8-6-2-5-7-3
4	286.5016514	90	20000	1-4-8-6-2-5-7-3
4.4	**286.5016514**	**90**	**5230**	

Table C-1 : Simulated Annealing Solutions for 8 Cities

Time	Cost	Seed	Temperature	Solution
14	401.6919473	100	100	1-5-11-8-12-6-2-4-10-9-7-3
14	401.6919473	100	100	1-5-11-8-12-6-2-4-10-9-7-3
14	401.6919473	100	100	1-5-11-8-12-6-2-4-10-9-7-3
14	401.6919473	100	100	1-5-11-8-12-6-2-4-10-9-7-3
14	401.6919473	100	100	1-5-11-8-12-6-2-4-10-9-7-3
14	401.6919473	100	100	1-5-11-8-12-6-2-4-10-9-7-3
14	401.6919473	100	100	1-5-11-8-12-6-2-4-10-9-7-3
14	401.6919473	100	100	1-5-11-8-12-6-2-4-10-9-7-3
14	401.6919473	100	100	1-5-11-8-12-6-2-4-10-9-7-3
14	401.6919473	100	100	1-5-11-8-12-6-2-4-10-9-7-3
14	**401.6919473**	**100**	**100**	

Table C-2 : Simulated Annealing Solutions for 12 Cities

Time	Cost	Seed	Temperature	Solution
14	502.7187932	100	20000	1-7-13-15-2-8-9-14-4-11-6-5-12-10-3
14	502.7187932	100	20000	1-7-13-15-2-8-9-14-4-11-6-5-12-10-3
14	502.7187932	100	20000	1-7-13-15-2-8-9-14-4-11-6-5-12-10-3
14	502.7187932	100	20000	1-7-13-15-2-8-9-14-4-11-6-5-12-10-3
14	502.7187932	100	20000	1-7-13-15-2-8-9-14-4-11-6-5-12-10-3
14	502.7187932	100	20000	1-7-13-15-2-8-9-14-4-11-6-5-12-10-3
14	502.7187932	100	20000	1-7-13-15-2-8-9-14-4-11-6-5-12-10-3
14	502.7187932	100	20000	1-7-13-15-2-8-9-14-4-11-6-5-12-10-3
14	502.7187932	100	20000	1-7-13-15-2-8-9-14-4-11-6-5-12-10-3
14	502.7187932	100	20000	1-7-13-15-2-8-9-14-4-11-6-5-12-10-3
14	**502.7187932**	**100**	**20000**	

Table C-3 : Simulated Annealing Solutions for 15 Cities

Time	Cost	Seed	Temperature	Solution
22	614.0940825	60	1000	1-9-3-6-18-19-8-2-5-20-11-13-15-4-10-17-7-14-12-16
22	614.0940825	60	1000	1-9-3-6-18-19-8-2-5-20-11-13-15-4-10-17-7-14-12-16
22	614.0940825	60	1000	1-9-3-6-18-19-8-2-5-20-11-13-15-4-10-17-7-14-12-16
22	614.0940825	60	1000	1-9-3-6-18-19-8-2-5-20-11-13-15-4-10-17-7-14-12-16
22	614.0940825	60	1000	1-9-3-6-18-19-8-2-5-20-11-13-15-4-10-17-7-14-12-16
22	614.0940825	60	1000	1-9-3-6-18-19-8-2-5-20-11-13-15-4-10-17-7-14-12-16
22	614.0940825	60	1000	1-9-3-6-18-19-8-2-5-20-11-13-15-4-10-17-7-14-12-16
22	614.0940825	60	1000	1-9-3-6-18-19-8-2-5-20-11-13-15-4-10-17-7-14-12-16
22	614.0940825	60	1000	1-9-3-6-18-19-8-2-5-20-11-13-15-4-10-17-7-14-12-16
22	614.0940825	60	1000	1-9-3-6-18-19-8-2-5-20-11-13-15-4-10-17-7-14-12-16
22	**614.0940825**	**60**	**1000**	

Table C-4 : Simulated Annealing Solutions for 20 Cities

Time	Cost	Seed	Temperature	Solution
43	807.9037467	90	10000	1-21-24-7-12-9-17-2-19-3-16-11-10-8-4-18-6-13-14-23-20-15-25-22-5
43	807.9037467	90	10000	1-21-24-7-12-9-17-2-19-3-16-11-10-8-4-18-6-13-14-23-20-15-25-22-5
43	807.9037467	90	10000	1-21-24-7-12-9-17-2-19-3-16-11-10-8-4-18-6-13-14-23-20-15-25-22-5
43	807.9037467	90	10000	1-21-24-7-12-9-17-2-19-3-16-11-10-8-4-18-6-13-14-23-20-15-25-22-5
43	807.9037467	90	10000	1-21-24-7-12-9-17-2-19-3-16-11-10-8-4-18-6-13-14-23-20-15-25-22-5
43	807.9037467	90	10000	1-21-24-7-12-9-17-2-19-3-16-11-10-8-4-18-6-13-14-23-20-15-25-22-5
43	807.9037467	90	10000	1-21-24-7-12-9-17-2-19-3-16-11-10-8-4-18-6-13-14-23-20-15-25-22-5
43	807.9037467	90	10000	1-21-24-7-12-9-17-2-19-3-16-11-10-8-4-18-6-13-14-23-20-15-25-22-5
43	807.9037467	90	10000	1-21-24-7-12-9-17-2-19-3-16-11-10-8-4-18-6-13-14-23-20-15-25-22-5
43	807.9037467	90	10000	1-21-24-7-12-9-17-2-19-3-16-11-10-8-4-18-6-13-14-23-20-15-25-22-5
43	**807.9037467**	**90**	**10000**	

Table C-5 : Simulated Annealing Solutions for 25 Cities