Search and Optimisation TSP Report

**By:** Faisal Abdul-Fattah

**Word count for the main body (from abstract up to references):** 2649

Contents

[Abstract: 3](#_Toc186208815)

[Introduction: 3](#_Toc186208816)

[Scope: 3](#_Toc186208817)

[Chosen methodology for comparing algorithms: 3](#_Toc186208818)

[Comparing algorithms: 4](#_Toc186208819)

[Comparing simulated annealing to the genetic algorithm: 6](#_Toc186208820)

[Problem instances: 7](#_Toc186208821)

[Improving algorithm: 8](#_Toc186208822)

[Best route found 8](#_Toc186208823)

[Results compared to existing literature 9](#_Toc186208824)

[Legal and ethical issues: 9](#_Toc186208825)

[Conclusions 10](#_Toc186208826)

[References (alphabetical order): 10](#_Toc186208827)

# Abstract:

This study aimed to test and compare various algorithms, some single-solution driven and some population-driven, when solving the Traveling Salesman Problem(TSP). The dataset had 50 cities total. The comparison of the algorithm started with the cost (distance), after that the efficiency, computational complexity, scalability and robustness were compared as well. The best algorithm identified was the simulated annealing algorithm.

# Introduction:

The Traveling Salesman Problem is the most famous and most investigated combinatorial problem in the world (Bresson and Laurent, 2021). For this study, 8 algorithms were tested. The project entailed going through every city once before returning to the initial city. This report will document the methodology, findings, analysis, and ethics of the project.

# Scope:

The scope of this project was limited to testing a few single-solution driven solutions and only one population-driven solution and then comparing them.

The scope was limited to 50 cities.

# Chosen methodology for comparing algorithms:

1) The most important aspect is the cost/route distance.

2) Efficiency. This was be measured using two metrics:

-The time taken. Measures the running time of the algorithm. Shows the convergence speed, which is time an algorithm takes to approach a solution (Barzilai, 2020).

-The memory usage. Note that memory used can also give an idea of the space complexity (Soni Upadhyay, 2024).

Measuring the time and memory needed by an algorithm to reach a solution shows the algorithmic efficiency (Huang, Lai and Cheng, 2009).

3) Computational complexity. This measures the resources that an algorithm uses when it runs, such as time and memory, so similar to the efficiency (‘computational complexity | Definition & Facts | Britannica’, n.d.).

4) The scalability. Does the algorithm show a noticeable drop in its performance when the problem size is raised by increasing the number of cities? This has been measured this through plotting graphs of the time taken and the route distance of the TSP problem on 10, 20, 30, 40, and then 50 cities. If the times and distances change linearly or logarithmically as city number is increased, this shows good scalability. If they increase exponentially/quadratically, this shows that the algorithm shows a noticeable drop in performance as city number in increased (Avish Saini, 2024).

5) Robustness. Does the algorithm perform well using different problem conditions, such as varying starting conditions or parameters? Will it consistently find the optimum solution? Will the results vary greatly when we change the conditions (Mastoi Week 3 Presentation, 2024)? To test the robustness, the time and distance taken when running the algorithms using 5 different starting points were measured. Then, these results were plotted. Subsequently, it was identified whether the algorithm reached the optimal distance/close to it constantly or not. Furthermore, the mean distance was measured. This was done so it can be studied whether the mean distance is close to the optimal distance (meaning the algorithm constantly achieves distances close to the optimal distance). The mean time was measured too to see if it varies a lot with the different starts. Furthermore, to measure the spread/variability of the results on a deeper level, the standard deviation was measured. To conceptualise the standard deviation/spread, a measure called the coefficient of variation was calculated which equates to (standard deviation/mean)\*100. Generally, lower than 5% is good meaning low variability, over 10% is bad as it means the results vary a lot with different runs (Zady, 2023).

# Comparing algorithms:

**1) Single solution-driven search algorithms**

**Scalability:**

A screenshot of a computer

Description automatically generated

Figure -Scalability comparison

**Robustness:**

A graph on a paper

Description automatically generated

Figure -Robustness comparison

A graph on a white sheet

Description automatically generated

Figure -Robustness comparison

A graph of a graph

Description automatically generated with medium confidence

Figure -Robustness comparison

Why use the mean distance over multiple runs? If the distance when running it once is used, it is not a very good indicator of performance because this distance can change drastically the next time the code is ran.

The 3 algorithms with the lowest cost were in order: Simulated annealing, Iterated local search, and Random Restart Hill Climbing. So:

Comparison:

* Distance/cost -> Best in order: Simulated annealing then Iterated local search and then Random Restart Hill Climbing.
* Efficiency -> Simulated annealing is the slowest of the algorithms, and uses the highest memory, therefore the least efficient. Random restart the most efficient, while iterated local search is in the middle.
* Computational complexity -> Same order as efficiency for the same reasons.
* Scalability -> Simulated annealing has bad time scalability due to being exponential, and good distance scalability. The other two algorithms have good scalability for both time and distance.
* Robustness -> Simulated annealing has good robustness all around, Random restart good distance and bad time robustness, while iterated local search has average distance robustness performance and bad time robustness performance.
* So, which is the best overall? The best is simulated annealing. This is because the most important aspect of comparison is the cost/distance, and simulated annealing performs best here. Furthermore, it is the most robust. Its downside is that it is time and memory consuming. It has bad time scalability, efficiency and computational complexity. But realistically the mean time is 443 seconds which is about 7.5 minutes, which is not that high. And the wait is worth it as it gives an excellent route distance value. And the memory of 0.5391 MiB is realistically not a problem for most computers. The advantages of this algorithm certainly outweigh the disadvantages and so it was picked.

However, just to validate this solution, a statistical t-test hypothesis test between the two best algorithms, simulated annealing and iterated local search, was done to validate how statistically significant the difference in the distance and time is between the two algorithms. 5 different runs were used for each algorithm.

A screenshot of a graph

Description automatically generated

Figure - Single solution driven algorithms t-test

1) Null hypothesis:

* There is no significant difference in the average distances or times of the two algorithms.

2) Significance level.

* Significance Level (α) is: 0.05 (5%).

3) The p-value and decision rule of the test.

* P-value for the distance is: p= 0.0240
* Decision rule: If the p-value is less than the significance level (α), reject the null hypothesis. In this case it is less than 0.05, so reject the null hypothesis. There is a significant difference in the average distances between the two algorithms.
* P-value from the t-test for the time is p= 0.0000
* Decision rule: p<0.05, will reject the null hypothesis. There is a significant difference in the average times between the two algorithms.

4) Conclusion and interpretation.

* Conclusion: There is sufficient evidence that suggests that there is a significant difference in the average distances between the two algorithms. So, using simulated annealing significantly lowers the cost.
* Conclusion: There is sufficient evidence that suggests that there is a significant difference in the average times between the two algorithms. Iterated local search significantly faster.
* Therefore, this t-test further validates the results. Even though iterated local search is significantly faster, in the end the most important aspect is the cost and simulated annealing provides significantly lower cost and so it is better.

# Comparing simulated annealing to the genetic algorithm:

**Scalability:**

**A screenshot of a computer

Description automatically generated**

Figure -Scalability comparison

**Robustness:**

**A graph of different types of graphs

Description automatically generated with medium confidence**

Figure -Robustness comparison

* This comparison was very easy. The simulated annealing performs better than or similar to the genetic algorithm in every single aspect. Better cost/distance, efficiency, and computational complexity, and similar scalability and robustness. Therefore, the final chosen algorithm was simulated annealing.

Statistical test:

A screenshot of a graph

Description automatically generated

Figure -Simulated annealing vs genetic algorithm t-test

To compare these 2 computationally expensive algorithms, only 3 different runs were done for both algorithms so that the computational cost is not too high.

In this case, from the bar plots, simulated annealing provides a slightly higher mean distance, which is not a problem YET. The results in figure 6 showed that the mean distance for both algorithms was very close. What is important in a t-test is if the performance is SIGNIFICANTLY better.

1) Null hypothesis:

Same as before.

2) Significance level

* Significance Level (α) is: 0.05 (5%).

3) The p-value and decision rule of the test.

* P-value for the distance is: p= 0.5099
* Decision rule: In this case it is not less than 0.05, so fail to reject the null hypothesis. There is no significant difference in the average distances between the two algorithms.
* P-value from the t-test for the time is p= 0.0001
* Decision rule: p<0.05, will reject the null hypothesis. There is a significant difference in the average times between the two algorithms.

4) Conclusion and interpretation.

* Conclusion: There isn't sufficient evidence that suggests that there is a significant difference in the average distances between the two algorithms. So, using either algorithm will not significantly affect the cost.
* Conclusion: There is sufficient evidence that suggests that there is a significant difference in the average times between the two algorithms. Simulated annealing significantly faster.
* Therefore, this t-test further validates that simulated annealing is better. It provides a significantly faster result with a cost that is similar (not significantly different) to the genetic algorithm.

# Problem instances:

(Abdulkarim and Alshammari, 2015) compared the algorithms at different numbers of cities problem instances. They used 20, 100, and 1000 cities.

Furthermore, (Pan *et al.*, 2023) also used problem instances of different city numbers (nodes). They tested a large-scale TSP, using instances with 1000, 2000, 5000, and 1000 nodes.

Therefore, taking inspiration from these examples in literature, the performance of the algorithms was compared at different city/node numbers below.

What if:

1) A longer run-time is available.

Then simulated annealing or genetic algorithm should be chosen (statistically they give similar results as the t-test showed).

2) 10 cities.

There are many algorithms with the same route distance using 10 cities, but simple hill climbing should be chosen. This is because it is the most efficient, least computationally expensive one.

3) 20 cities.

Simple hill climbing, for the same reason as 10 cities. Why does simple hill climbing work as well as the complex algorithms at these low scale problems? Small problem size with a lower number of nodes and local optima; less rugged. Less chance of getting stuck at local optima.

4) 30 cities.

Simulated annealing, genetic algorithm, and tabu search give the same values for the distance here. However, tabu search should be picked as it’s the most efficient, least computationally expensive one. Why are these more complex algorithms better as city number in increased? This is because these 3 methods can escape local minima more effectively than the other algorithms. Which is important because the local minima number will increase when the number of cities increases. For example, tabu search can escape local optima through selecting solutions that are non-improving and by not returning to old solutions

5) 40 cities.

Simulated annealing, iterated local search, and genetic algorithm give the same values for the distance here. However, iterated local search should be picked as it’s the most efficient, least computationally expensive one.

6) 50 cities.

As explained before, simulated annealing is the best choice here. But why? Simulated annealing can escape local minima. It has a probabilistic acceptance of solutions that may not be better because it means the algorithm can explore a wider search space. Simulated annealing also balances exploration and exploitation well (Henderson, Jacobson and Johnson, n.d.).

7) Short run-time is necessary, but still need a good result even if it is not the best one.

Random restart balances a small run-time, good solution, high efficiency, and good scalability. Why does random restart perform better than simple hill climbing at 50 cities? This is because the search space is rugged, with multiple local optima. Random restarts help the programme leave local optima. It gives multiple attempts at a better solution through random restarts.

# Improving algorithm:

Often when code for the algorithm is run, the route distance reached becomes constant for a big block of the iterations. Code was added that stops the algorithm if it that happens. If the algorithm does not improve the cost for 400 iterations in a row, the algorithm will stop. This value was obtained through trial and error. Through testing, it was proven that typically if values do not improve for 400 iterations straight, then they will never improve up until the max iterations. So, the algorithm now stops if the algorithm gets stuck like this and hence improves efficiency and computational complexity.

A screen shot of a computer

Description automatically generated

Figure -Improved algorithm

# Best route found

The best route for one of the simulated annealing runs was saved in an external file:

A screenshot of a spreadsheet

Description automatically generated

Figure -Best route

A screenshot of a table

Description automatically generated

Figure -Best route

A graph with blue dots and lines

Description automatically generated

Figure -Best route visualised

# Results compared to existing literature

(Pan, 2015)compared algorithms in terms of the distance, time, and number of iterations. They tested a nearest neighbour algorithm, genetic algorithm, and a greedy heuristic algorithm. They simulated the algorithms using MATLAB. Their final verdict was that the greedy heuristic algorithm reaches the closest value to the optimal value and so this is their best algorithm. This result is different to the result of this current study, since simulated annealing is specifically a metaheuristic, not just a heuristic.

Which algorithm has the highest total publications about being used in the TSP? (Bladimir Toaza and Domokos Esztergár-Kiss, 2023) analysed nearly 50 years of research and demonstrated how the top 5 are genetic algorithm, ant colony optimisation, simulated annealing, particle swarm optimisation, and tabu search. Therefore, the chosen algorithm for the current project (simulated annealing) has historically been used many times in research to solve TSP.

It is also interesting to research newer literature from this year. (Hossain and Yılmaz Acar, 2024) compared some of the classic algorithms such as genetic algorithm and simulated annealing and some newer algorithms such as one called ‘Ant Colony Optimization’. They found that the newer ones offer improved results in convergence, scalability, as well as computation time than the classical methods. This shows that even though simulated annealing works well, there is always room for improvement.

# Legal and ethical issues:

Code used: Various codes from sources online were used as inspiration for this current project. All of them were either from google websites available publicly, or from GitHub, a website where people can upload their codes. There were two main sources. One of them was (Nour Oulad Moussa, 2023). He uploaded the code publicly online with no identification anywhere on the page that the code should not be used by others. No mentions of licensing or restrictions on using the code. The second source of code was (Valdecy, 2021). He has stated as shown in figure 13 that the software is free, and under the GLU General Public License (GPL). This means it is an open-source software. It is open for everyone to see and use. With this license, the access, modification, and even distribution of this software, are permitted (Viroux, 2023).

A screenshot of a computer software update

Description automatically generated

Figure -License on the code used

Dataset used: The dataset used was not form a private source. No license needed. It was provided to the students for academic use.

If software was used without the appropriate rights (piracy), punishments can include a hefty fine or serving prison time. It all depends on the severity of the code pirated. Risk of being sued for illegal use of intellectual property of the company (‘Essential guide to using software legally’, n.d.).

But what if these resources were being used for commercial use?

Using open-source software for commercial reasons is acceptable and legal, if the owners give permission. Sometimes if open-source software is used commercially, the business may be forced to notify users that open-source code was used and what licenses apply to them. For example, when talking specifically about GPL licensed code like the one this current project used, one of the rules is that if this code is used or modified then it must also be licensed under the GPL and distributed to others under these same terms (figure 13)(Viroux, 2023).

# Conclusions

This project analysed 8 algorithms total and tested the performance of each of them at solving the TSP. The algorithms were compared in terms of computational complexity, efficiency, distance/cost, scalability, and robustness. The best algorithm when using 50 cities was simulated annealing. Genetic algorithm was another great choice.

# References (alphabetical order):

‘computational complexity | Definition & Facts | Britannica’ (no date) *www.britannica.com* [online]. Available from: https://www.britannica.com/topic/computational-complexity [Accessed 20 December 2024].

‘Essential guide to using software legally’ (no date) *www.techdonut.co.uk* [online]. Available from: https://www.techdonut.co.uk/business-software/software-licensing/essential-guide-to-using-software-legally [Accessed 20 December 2024].

Abdulkarim, H. and Alshammari, I. (2015) Comparison of Algorithms for Solving Traveling Salesman Problem Comparison of Algorithms for Solving Traveling Salesman Problem. *International Journal of Engineering and Advanced Technology (IJEAT)*. (6), pp. 2249–8958. [Accessed 16 December 2024].

Avish Saini (2024) Here’s how you can evaluate the scalability of an algorithm effectively. Linkedin.com. 2024 [online]. Available from: https://www.linkedin.com/advice/0/heres-how-you-can-evaluate-scalability-algorithm-effectively-efuif [Accessed 26 November 2024].

Barzilai, J. (2020) On neural-network training algorithms. *Elsevier eBooks* [online]. pp. 307–313. Available from: https://www.sciencedirect.com/topics/computer-science/speed-convergence [Accessed 20 December 2024].

Bladimir Toaza and Domokos Esztergár-Kiss (2023) A review of metaheuristic algorithms for solving TSP-based scheduling optimization problems. *Applied Soft Computing* [online]. 148, pp. 110908–110908.

Bresson, X. and Laurent, T. (2021) *The Transformer Network for the Traveling Salesman Problem* *arXiv.org*. 2021 [online]. Available from: https://arxiv.org/abs/2103.03012 [Accessed 16 December 2024].

Henderson, D., Jacobson, S.H. and Johnson, A.W. (no date) The Theory and Practice of Simulated Annealing. *International Series in Operations Research & Management Science* [online]. pp. 287–319. [Accessed 17 December 2024].

Hossain, M.A.A. and Yılmaz Acar, Z. (2024) Comparison of New and Old Optimization Algorithms for Traveling Salesman Problem on Small, Medium, and Large-scale Benchmark Instances. *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi* [online]. 13 (1), pp. 216–231. Available from: https://dergipark.org.tr/en/pub/bitlisfen/issue/83698/1380086 [Accessed 20 December 2024].

Huang, C.-Y. (Ric), Lai, C.-Y. and Cheng, K.-T. (Tim) (2009) Fundamentals of algorithms. *Electronic Design Automation* [online]. pp. 173–234.

Margaret Rouse (2019) What is Computational Complexity? - Definition from Techopedia Techopedia.com. 6 February 2019 [online]. Available from: https://www.techopedia.com/definition/18466/computational-complexity [Accessed 20 December 2024].

Mastoi Week 3 Presentation, Q.-U.-A. (2024) Week 8 Lecture Uwe.ac.uk. 2024 [online]. Available from: https://blackboard.uwe.ac.uk/ultra/courses/\_365434\_1/cl/outline [Accessed 24 November 2024].

Nour Oulad Moussa (2023) *Exploring TSP with Hill Climbing algorithms - Nour Oulad Moussa - Medium* *Medium*. 4 October 2023 [online]. Available from: https://medium.com/@nour.oulad.moussa/exploring-tsp-with-hill-climbing-algorithms-9c4762ecd483 [Accessed 16 December 2024].

Pan, X., Jin, Y., Ding, Y., Feng, M., Zhao, L., Song, L. and Bian, J. (2023) H-TSP: Hierarchically Solving the Large-Scale Traveling Salesman Problem. *Proceedings of the AAAI Conference on Artificial Intelligence* [online]. 37 (8), pp. 9345–9353. Available from: https://www.microsoft.com/en-us/research/uploads/prod/2023/02/8643.PanX\_.pdf [Accessed 20 December 2024].

Soni Upadhyay (2024) Time and Space complexity in Data Structure - Ultimate Guide Simplilearn.com. 3 December 2024 [online]. Available from: https://www.simplilearn.com/tutorials/data-structure-tutorial/time-and-space-complexity#what\_is\_time\_complexity [Accessed 16 December 2024].

Valdecy (2021) GitHub - Valdecy/pyCombinatorial: A library to solve the TSP (Travelling Salesman Problem) using Exact Algorithms, Heuristics and Metaheuristics : 2-opt; 2.5-opt; 3-opt; 4-opt; 5-opt; 2-opt Stochastic; 2.5-opt Stochastic; 3-opt Stochastic; 4-opt Stochastic; 5-opt Stochastic; Ant Colony Optimization; Bellman-Held-Karp Exact Algorithm; Branch & Bound; BRKGA (Biased Random Key Genetic Algorithm); Brute Force; Cheapest Insertion; Christofides Algorithm; Clarke & Wright (Savings Heuristic); Concave Hull Algorithm; Convex Hull Algorithm; Elastic Net; Extremal Optimization; Farthest Insertion; Genetic Algorithm; GRASP (Greedy Randomized Adaptive Search Procedure); Greedy Karp-Steele Patching; Guided Search; Hopfield Network; Iterated Search; Karp-Steele Patching; Multifragment Heuristic; Nearest Insertion; Nearest Neighbour; Random Insertion; R… GitHub. 2021 [online]. Available from: https://github.com/Valdecy/pyCombinatorial?tab=readme-ov-file [Accessed 21 November 2024].

Viroux, G. (2023) *Understanding Software Licenses: Open-Source vs. Commercial* *Medium*. 28 March 2023 [online]. Available from: https://glenn-viroux.medium.com/understanding-software-licenses-open-source-vs-commercial-ad365b160909 [Accessed 16 December 2024].

Zady, M.F. (2023) Z-4: Mean, Standard Deviation, And Coefficient Of Variation - Westgard Westgard.com. 2023 [online]. Available from: https://westgard.com/lessons/z-stats-basic-statistics/lesson34.html [Accessed 17 December 2024].