The Travelling Salesman Problem, Comparing the Classical and Quantum Computing Approach

2341197 - Aidan Rabinowitz, 1924564 - Jess Dworcan

***Abstract*—This report explores solving the Traveling Salesman Problem, through different classical computing algorithms, and using the Quantum Approximate Optimization Algorithm on IBM quantum hardware. The classical methods are compared with the quantum, and the feasibility and effectiveness of the different methods are evaluated. The classical methods are shown to outperform the quantum method simply due to the practical limitations of using an IBM quantum computer. Simulated Annealing utilising the nearest neighbour method as the initial tour, is found to be the most accurate and efficient solution.**

I. Introduction

THE Travelling Salesman Problem (TSP) is an optimization problem which aims to find the shortest route an individual must travel between a group of points (nodes), visiting each point once and returning to the start. Finding an algorithm which finds the ideal solution to this problem is classified as NP-Hard, so there is no optimal algorithm. Heuristic algorithms are used to find the best solutions available with the given computation power[1]. This report examines several classical and one quantum algorithm, chosen to find the ideal solution to the TSP, and records the time taken for an Intel® Core™ i7-10870H Processor 2.2 GHz processor to complete the classical algorithms, and for the IBM Eagle r3 processor to complete the quantum algorithm. The relevant code containing these algorithms is found in the github repository at https://github.com/jessdcan/QC_Project_TSP.

II. Background

The TSP is a combinatorial graph theory problem which has far-reaching influence. Almost every industry on earth includes some form of transport, which could be optimised using TSP algorithms. From logistics companies optimising shipping routes to minimise fuel consumption and maximise sales, to circuit board designers, to optimising DNA sequencing[2]. All of these practical problems can be simplified into an object (salesman) which must travel to a certain combination of nodes. The object can take several paths, but there exists only one most efficient path (except in the very unlikely case that two or more minimum paths exist), and the object's overall efficiency will greatly improve if it knows the shortest path to take.

This problem in this report is abstracted to use numbered nodes, joined with lines, where the nodes to which the salesman must travel are red balls with numbers for classification, and whichever route is determined as the most optimal by different algorithms, is graphically displayed in the relevant code.

Finding the ideal solution to the TSP on a classical computer is famously difficult, for a large number of nodes, because of its $O(n!)$ complexity, where n is the number of nodes.

A. Assumptions for the TSP

- Distances are commutative
- Distances are non-negative
- Distances satisfy the triangle inequality, and exist within a two-dimensional plane.
- Nodes have no weight

III. Classical TSP Solutions

There are many different approaches to solving the TSP using classical computers. This report explores four such methods namely, Brute Force, Nearest Neighbor (NN), Simulated Annealing (SA) with a random initial tour, and an adaptation of SA using NN as the initial tour.

A. Brute Force

The brute force approach is the least computationally efficient classical algorithm for solving the TSP. This method involves calculating every possible route to traverse between all nodes to find the shortest route. For ten nodes, the brute force method needs $(10-1)!$ or $362,880$ iterations to compute the TSP[3]. Heuristic algorithms have thus been found due to the inefficiency of the brute force approach. The brute force method finds the ideal solution, but its time complexity increases so quickly that heuristic algorithms are favoured for practical use.

B. Nearest Neighbor

The Nearest Neighbour algorithm works by iteratively building a tour by selecting the nearest unvisited neighbour to the current node at each step[4]. This process continues until all nodes have been visited, forming a complete tour. The algorithm repeatedly selects the nearest unvisited neighbour until all nodes are visited. While straightforward, this algorithm doesn't guarantee an optimal solution, as it may get stuck in local optima[3]. However, it is computationally efficient and easy to implement, making it a popular choice for solving small to moderate-sized instances of optimization problems.

C. Simulated Annealing

Simulated annealing is an optimization algorithm that mimics the process of annealing in metallurgy. Initially, the algorithm explores the solution space randomly. It accepts node moves that improve the solution. However, it also occasionally accepts worse moves based on a probability determined by the temperature parameter. This randomness allows the algorithm to escape local optima and explore different regions of the

solution space. Over time, the temperature decreases, allowing fewer bad moves to happen. Eventually, the algorithm converges to a low-energy state, representing a good solution to the optimization problem. This algorithm is documented as taking longer to compute solutions than other popular optimisation algorithms[5].

D. Simulated Annealing with Nearest Neighbor as Initial Path

As opposed to choosing a random path to start the annealing process from, the simulated annealing process can be improved upon by using the solution of the NN approach as an initial start path. This combined approach is effective at finding more accurate approximations of the TSP[5]. The class SA_NN_Class contains the functionality that performs simulated annealing with the initial solution of the nearest neighbour[6]. The functions of the class are called at the end of the main file in the relevant code.

IV. Quantum TSP Solution

Heuristic algorithms attempt to solve this important problem, with their best assumptions of solutions, but Quantum Computing (QC) can hold the optimised solution by exploiting quantum phenomena like superposition and entanglement. Essentially, the foundation of QC is the quantum bit (qubit), which has the ability to store statistical data about several states at once, only collapsing into a set state upon observation. Marinescu (2005) explains how QC allows "an exponential increase in parallelism requires only a linear increase in the amount of space needed"[1]. Van Dam et al.[2] and Farhi et al.[7] explain how this theoretically allows for Adiabatic Quantum Computing, using quantum annealing, wherein the ground state of a complex Hamiltonian, representing the energy of a quantum system, is evolved from a simpler Hamiltonian, slowly altering the first Hamiltonian to keep the system close to the ground state. The ground state in the TSP is achieved when the lowest eigenvalues of the Hamiltonian are found, which corresponds to an ideal solution i.e. the shortest distance for the TSP.
Quantum annealing is unachievable with the current circuit gate model Qiskit uses on IBM's quantum computers, because even a small error caused by mild temperature fluctuations above zero degrees Kelvin would cause the Hamiltonian to stray too far from its ground state to obtain true quantum annealing. This is why, until the hardware is improved, quantum approximations are the best way of finding the most ideal solutions to the TSP and other Quadratic Unconstrained Binary Optimization (QUBO) problems which can be solved using the ground state of a Hamiltonian.

A. Solving the TSP with a Quantum Algorithm

The next best thing after an ideal quantum algorithm, is an approximation quantum algorithm. These are often found as hybrid quantum-classical algorithms, because certain classical computing properties like storing results and formatting matrices are suited better to classical computers.

1) The Quantum Approximate Optimization Algorithm: The Quantum Approximate Optimization Algorithm (QAOA) is one such hybrid algorithm which is used to approximate a ground state Hamiltonian and is thus used to solve the TSP. This algorithm is designed to use a quantum circuit with unitary gates[7]. The algorithm takes advantage of the time-dependent Schrödinger equation $i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \hat{H}\Psi(\mathbf{r},t)$, because this describes the changing of a Hamiltonian over time. This algorithm uses trotterization to size the time increments of the growing system, with more increments providing a greater accuracy.

V. Results and Analysis

The TSP for three to eleven nodes is run on a classical computer and for three to five nodes on IBM's quantum Eagle r3 processor with 127 qubits. 3 nodes on IBM Cusco; 4 nodes on IBM Kalkata; 5 nodes on IBM Sherbrooke. These three quantum computers each have the same number of qubits and processor technology, but the layout of the qubits, and also the Error Per Layered Gate (EPLG) percentage differs for each. Ideally, one computer should be used to compare the different node solution times, but due to job number limitations, after initially testing several computers, testing became unavailable. Three sessions were run for each node count, to increase the accuracy of results. Due to the iterative nature of the QAOA solution, which sends a new request to the IBM server every time a new Hamiltonian energy is calculated, calculating the solution to six and more nodes (after running many jobs for 3,4 and 5 nodes) consistently resulted in a "Reduce time between submitting subsequent jobs in a session." error, hence the N/A values in Table I. The time taken and number of iterations needed for each node are shown in table I, comparing a classical brute force method to the QAOA quantum method. Further classical method results are tabulated in Table II in the appendix.

A. Classical Brute Force

The brute force method performed reasonably quickly for fewer than 10 nodes but becomes extremely slow as nodes are incremented slightly higher. Testing was stopped at 11 nodes, because the time taken for more results would be too greate above the 93s for 11 nodes. This method provides the ideal shortest route, as it examines every possible route, so it is helpful as a control group to measure how closely other algorithms reach an ideal solution.

B. Quantum QAOA

Each time the QAOA calculates the energy of the Hamiltonian and changes the matrix, it sends a job request to the relevant quantum computer. This request and the time taken for it to complete are recorded under the 'Jobs' tab in the IBM Quantum Platform. The number of jobs sent and the cumulative job time are recorded in Table 1. The times exclude the time spent queuing for an available machine, which was on average approximately three hours, after which each job was completed in succession until a solution was found, or

an error was thrown. Quantum length is omitted, because for 3,4 and 5 nodes, the ideal solution was found. Interestingly, this accuracy only presented when running 10000 shots on the quantum backend. For 100 and 1000 shots, nonsense data, sometimes showing random routes of random lengths, was found.

The QAOA algorithm run on a quantum computer has serious practical drawbacks, largely related to the number of iterations it requires: Firstly, the number of iterations required for each node is not predictable, because the starting parameter for the algorithm, x0, can be far off or close to the final answer, which can greatly change the number of iterations needed. Secondly, having a large number of iterations works theoretically, but on the shared IBM quantum computers, calling too many jobs results in errors, to prevent overuse. This disabled viable results for more than 5 nodes. Furthermore, n^2 qubits are required to solve n nodes, so if there was no restriction on the number of jobs possible, the physical limit for the 127 qubit machine would have been reached after 11 nodes. Transpilation would enable restructuring of the circuit built using qiskit, to suit the hardware, which may also decrease noise, but this would likely make the system even less efficient[8].

C. Nearest Neighbours

The Nearest Neighbours method of finding solutions to the TSP performed well overall, beating SA and SA with NN in the time taken to find a solution. This is likely due to the simplicity of the algorithm as it requires fewer operations to run. Sometimes the solution found would be far off from the ideal solution, due to unforeseen complications in that run of the algorithm, which depend on the initial structuring of the problem, like for node 7 in Table II, i.e. it can sometimes get stuck in local optima causing large jumps between nodes that increase path length.

D. Simulated Annealing with a Random Initial Tour (RIT)

This method shows reliable results, predictably increasing the time taken and the path length when it finds a solution. However, compared to the other classical methods explored, this solution is the least accurate classical heuristic algorithm, i.e. it produces the longest routes. It is also slower than the Nearest Neighbour algorithm, so it is determined as the worst classical algorithm in this report for the TSP.

E. Simulated Annealing with Nearest Neighbours Initial Tour (with NN)

This second approach to the initial tour generation for Simulated Annealing takes much longer to run, given that it has to generate code to find an initial Nearest Neighbour solution before it can begin simulating the annealing process, but it yields the most accurate ideal path lengths of all the algorithms in the classical domain. It improves upon the Nearest Neighbor algorithm by removing the inefficiency of getting stuck in local optima[3].

TABLE I. TIME AND ITERATIONS TAKEN TO SOLVE THE TSP WITH QUANTUM QAOA AND CLASSICAL BRUTE FORCE

Number of Nodes	Quantum Time (s)	Quantum Iterations	Classical Time (s)	Classical Iterations	Brute Force Length
3	540	60	0.0001	2	220
4	1020	90	0.0001	6	236
5	1421	170	0.001	24	271
6	N/A	N/A	0.001	120	279
7	N/A	N/A	0.259	720	279
8	N/A	N/A	0.210	5040	284
9	N/A	N/A	1.809	40320	286
10	N/A	N/A	7.715	362880	286
11	N/A	N/A	93.178	3628800	287

VI. FUTURE IMPROVEMENTS

The most obvious improvement would be regarding the implementation of quantum computers. Firstly, the quantum results should be generated for all nodes, and more quantum optimisation algorithms, like the Variational Quantum Eigensolver (VQE), should be used, for a complete comparison to the classical methods. Further investigation into an ideal quantum annealing algorithm should be done, which may involve designing new hardware to handle this process.

Hardware improvements would benefit future exploits. For example, exclusive access to one quantum computer for all jobs, so that one complete experiment can be run on the same quantum computer in the same state, within the time scope of the same calibration, which usually occurs daily[9].

VII. ACKNOWLEDGEMENTS

The QAOA and TSP functions are found in Qiskit tutorials, and the relevant logic and inspiration for using the QAOA to solve the TSP originated from these tutorials[10]. The logic for the code which accomplishes SA with NN as the initial solution is inspired by the GitHub user "chncyhn" [6]. ChatGPT was useful in fixing and improving the logic for the classical algorithms and was used to add time recording and fix errors.

It was also used to generate comments for some of the code. The comments in the SA_NN_Class file are attributed to the authors of this report.

VIII. DIVISION OF WORKLOAD

The workload was split so that one author focused solely on classical algorithms and one author on classical and quantum, because it was determined that having an understanding of the classical algorithms is important to compare the quantum. Both authors covered code functionality for brute force, NN and SA. Author 1924564 covered the SA with NN initial solution functionality. Author 2341197 covered the code functionality for the quantum algorithms. The quantum aspect of the report is written by author 2341197. The classical aspect of the report is written by author 1924564. Both authors collaborated with version control through Github, report editing and submission of the final solution.

IX. Conclusion

In conclusion, the practical drawbacks of current quantum computers make running a problem like the TSP very difficult, although if more resources, like a dedicated quantum computer, were allocated to it, it would perform much better than seen in this report. The classical algorithms which were examined performed much more efficiently, with the Nearest Neighbours algorithm excelling in time taken to solve the TSP, and Self Annealing with a Nearest Neighbours initial tour performed the most accurately when compared to the ideal brute force method.

The Self Annealing with Nearest Neighbours is determined to be the best way to solve the TSP, because although it was the slowest of all heuristic classical algorithms, it was the most accurate, and in a practical scenario like creating a shipping route for a logistics company, accurately estimating the quickest route will save more time overall.

REFERENCES

[1] D. C. Marinescu, "The promise of quantum computing and quantum information theory-quantum parallelism," in *19th IEEE International Parallel and Distributed Processing Symposium*, IEEE, 2005, 3–pp.

[2] W. Van Dam, M. Mosca, and U. Vazirani, "How powerful is adiabatic quantum computation?" In *Proceedings 42nd IEEE symposium on foundations of computer science*, IEEE, 2001, pp. 279–287.

[3] L. Daniells. "The travelling salesman problem – libby daniells." (), [Online]. Available: https://www.lancaster.ac.uk/stor-i-student-sites/libby-daniells/2020/04/21/the-travelling-salesman-problem/ (visited on 03/03/2024).

[4] G. Kizilateş and F. Nuriyeva, "On the nearest neighbor algorithms for the traveling salesman problem," in *Advances in Computational Science, Engineering and Information Technology: Proceedings of the Third International Conference on Computational Science, Engineering and Information Technology (CCSEIT-2013), KTO Karatay University, June 7-9, 2013, Konya, Turkey-Volume 1*, Springer, 2013, pp. 111–118.

[5] X. Geng, Z. Chen, W. Yang, D. Shi, and K. Zhao, "Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search," *Applied Soft Computing*, vol. 11, no. 4, pp. 3680–3689, 2011.

[6] chncyhn, *Simulated-annealing-tsp: Simulated Annealing algorithm to solve Travelling Salesmen Problem in Python*, https://github.com/chncyhn/simulated-annealing-tsp/tree/master, [Accessed 05-04-2024].

[7] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

[8] E. Wilson, S. Singh, and F. Mueller, "Just-in-time quantum circuit transpilation reduces noise," in *2020 IEEE international conference on quantum computing and engineering (QCE)*, IEEE, 2020, pp. 345–355.

[9] *About calibration jobs — IBM Quantum Administration — docs.quantum.ibm.com*, https://docs.quantum.ibm.com/admin/calibration-jobs, [Accessed 05-04-2024].

[10] *Max-cut and traveling salesman problem - qiskit optimization 0.6.1*. [Online]. Available: https://qiskit-community.github.io/qiskit-optimization/tutorials/06_examples_max_cut_and_tsp.html (visited on 03/03/2024).

APPENDIX

TABLE II. TIME TAKEN AND LENGTHS FOUND FOR DIFFERENT NODE NUMBERS IN THE TSP

Number of Nodes	NN Time (s)	NN Length (units)	SA RIT Time (s)	SA RIT Length (units)	SA with NN Time (s)	SA with NN Length
3	0.003	202	0.009	202	0.0366	202
4	0.0002	258	0.0136	236	0.0432	236
5	0.0011	271	0.0235	271	0.0557	271
6	0.0002	279	0.0389	279	0.0632	279
7	0.0002	317	0.0159	279	0.0626	279
8	0.001	284	0.0196	284	0.0692	284
9	0.001	286	0.0261	327	0.0796	285
10	0.0002	286	0.0139	348	0.0818	287
11	0.0009	287	0.0239	355	0.0921	289