Evaluating Quantum Annealing for the Travelling Salesman Problem

Evan Maxted
School of Computer Science
Carleton University
Ottawa, Canada
evanmaxted@cmail.carleton.ca

***Abstract*—The travelling salesman problem (TSP) is a well-known NP-hard problem with practical significance in routing and optimization. This paper considers the ability of quantum annealing to solve the TSP efficiently. We examine the encoding strategies surrounding the Hamiltonian operator and its application for quantum optimization algorithms. The complexity and feasibility of quantum annealing are compared to the classical algorithms of Bellman-Held-Karp and Christofides' algorithm. Our testing highlights that while quantum annealing shows theoretical promise, the unbounded approximations, qubit requirements, and annealing times are too complex for large instances and produce poor results. As such, classical approximation algorithms currently offer more reliable and scalable performance for TSP, though future work in hybrid models and bounded quantum approximations may change that.**

***Index Terms*—Hamiltonian operators, Ising models, NP-hard, quantum algorithms, quantum annealing, travelling salesman problem (TSP)**

I. Introduction

The travelling salesman problem (TSP) is one of the most well-studied NP-hard problems, with significant implications for fields such as routing and optimization. While classical algorithms have produced effective approximation methods for structured variants of the TSP, we ask whether quantum algorithms can offer improved performance.

This paper examines quantum annealing as the approach to solving the TSP. The goal of analyzing this method is to determine if quantum annealing is a satisfactory approach to solving the NP-hardness of the TSP.

We determine that although quantum annealing offers an interesting theoretical framework and some potential speedups, the approach remains mostly conceptual rather than practical. As it fails to find a valid path in many instances and produces unbounded approximations in others, there does not appear to be an advantage in choosing quantum computing over classical in this instance. The high qubit requirement and long annealing times make quantum annealing challenging to encode with current quantum capabilities. As a result, classical approximation algorithms currently provide more reliable and scalable solutions for the TSP.

The rest of this report is structured as follows. Section II introduces the key concepts concerning the TSP and quantum algorithms for the paper. Section III details classical computing approaches to solving the TSP. Section IV describes how quantum annealing is used for the TSP. Section V details the methods used to implement and test the algorithm. Testing results are given in Section VI. Section VII discusses the work of the paper. The limitations of the paper are discussed in Section VIII. Future work is proposed in Section IX. We conclude with Section X.

II. Background

A. Complexity Classes

Decision problems solved efficiently in polynomial time are in the complexity class P. Decision problems that can be yes-verified in polynomial time but not necessarily solved in polynomial time are in the class NP [15].

NP-hard is the set of all problems (decision and optimization) that are at least as hard as the hardest problems in NP. This contains the hardest optimization problems–such as the TSP–that are not verifiable in polynomial time. A problem in NP and NP-hard is in the class NP-complete, meaning NP-complete problems are the hardest NP problems but are still yes-verifiable in polynomial time [1].

B. Travelling Salesman Problem

The travelling salesman problem has two types: optimization and decision. Both types of the TSP concern the cost of a path that visits every node in the graph exactly once that starts and finishes at the same node–also known as a Hamiltonian cycle [8].

The optimization problem is finding the minimum cost cycle through all nodes. The optimization TSP is an NP-hard problem since there is no efficient way to verify a tour has minimum cost. The decision problem is asking if there is a tour with weight at most k. The decision TSP is an NP-complete problem since a provided solution can have its weight verified as valid in polynomial time. Without the conjecture P=NP, neither problem can be solved in polynomial time.

C. Hamiltonians

In quantum computing, the Hamiltonian operator H is used for solving optimization problems. The Hamiltonian corresponds to the energy of a system, and its ground state–the state with minimum energy–represents the optimal solution to the problem [12].

Hamiltonians can be designed to represent the cost functions and constraints of combinatorial optimization problems, allowing solutions to be found through energy minimization. By constructing a Hamiltonian whose ground state corresponds to the solution of a problem, we can use quantum algorithms to find this ground state and solve the problem.

D. Transverse Field Ising Models

Lucas [12] gives the Hamiltonians

$$H_D = -\sum_{i=1}^{N} \sigma_i^x$$

and

$$H_P = -\sum_{i<j} J_{ij}\sigma_i^z\sigma_j^z - \sum_{i=1}^{N} h_i\sigma_i^z\,,$$

where σ_i^x is the Pauli X gate and σ_i^z is the Pauli Z gate. h and J are the local fields and couplings respectively, which encode the problem of interest [3].

E. Quantum Adiabatic Evolution

Farhi et al. introduced an algorithm that applies the adiabatic theorem to solve combinatorial search problems [6]. This theorem states that if a quantum system starts in the ground state of an initial Hamiltonian $H(0)$ and the Hamiltonian evolves to a final Hamiltonian $H(T)$, the system will remain close to the ground state if the transformation is slow enough.

The algorithm evolves a time-dependent Hamiltonian $H(t)$, starting from an initial Hamiltonian $H(0)$ with a known, simple ground state. The final Hamiltonian $H(T)$ is encoded such that its ground state equals the solution to the problem.

The transformation of the time-dependent Hamiltonian is typically defined as the linear interpolation

$$H(t) = \left(1 - \frac{t}{T}\right) H(0) + \left(\frac{t}{T}\right) H(T)\,,$$

where t represents time and T is the total evolution time.

F. Quantum Annealing

Rajak et al. describe the quantum annealing method most commonly used to solve optimization problems [16]. A time-dependent Hamiltonian in an adiabatic system defines the system for quantum annealing:

$$H(t) = A(t)H_D + B(t)H_P\,.$$

$H(t)$ represents the system's total energy at time t, which reflects how optimal the current state is. H_D is the driver Hamiltonian, which has the trivial ground state and does not depend on the problem. H_P is the problem Hamiltonian, which encodes the optimization problem to be solved. $A(t)$ and $B(t)$ control the interpolation between H_D at the initial time and H_P at the final time.

Through quantum adiabatic evolution, the goal is for the system to stay in ground state through a slow transition between Hamiltonians. Ideally, H_P is reached and in the ground state corresponding to the optimal solution of the problem.

III. Classical Approaches

A. Deterministic Algorithms

The most simple algorithm for solving the TSP is the brute force algorithm. It involves comparing all possible permutations of cycles in the graph and taking the one with minimum cost [18]. Since all permutations must be explored to determine the minimum cost path, the runtime is $O(n!)$.

A dynamic programming algorithm was proposed by Bellman and by Held and Karp to solve the TSP in exponential $O(n^2 2^n)$ time [2, 11]. The algorithm works by finding the shortest paths on subgraphs and using them to eliminate the search space on larger subgraphs and paths.

The combination of exponential runtime and storage for storing all subgraph distances also makes this algorithm infeasible for large graphs [10].

B. Approximation Algorithms

A common method for efficiently solving NP-hard optimization problems is through approximation. Christofides gives a TSP approximation algorithm with a strict $\frac{3}{2}$ approximation ratio and $O(n^3)$ runtime [4]. Goodrich and Tamassia [7] detail the steps of the algorithm as follows:

1) Construct a minimum spanning tree of the graph G.
2) Construct a subgraph of G using only nodes with odd degrees in the minimum spanning tree.
3) Construct a minimum-weight perfect matching of the subgraph.
4) Add the edges from the perfect matching to the minimum spanning tree, allowing for duplicate edges.
5) Build an Eulerian circuit on the updated subgraph.
6) Convert the Eulerian circuit to a TSP tour by skipping repeated nodes.

For this algorithm to work, path lengths must form a metric space. This restriction ensures when skipping repeated nodes in the 6$^{\text{th}}$ step the triangle inequality guarantees the path length does not increase. The graph must also be complete to ensure an edge between the nodes.

IV. Quantum Annealing

A. Hamiltonians

Lucas [12] defines the Hamiltonian energy function for solving the TSP on a graph $G = (V, E)$ as follows:

$$\begin{aligned}
H &= H_A + H_B\,,\\
H_A &= A\sum_{i=1}^{n}\left(1 - \sum_{j=1}^{n} x_{ij}\right)^2\\
&\quad + A\sum_{j=1}^{n}\left(1 - \sum_{i=1}^{n} x_{ij}\right)^2\\
&\quad + A\sum_{(uv)\notin E}\sum_{i=1}^{n} x_{ui}x_{v(i+1)}\,,\\
H_B &= B\sum_{(uv)\in E} w(uv)\sum_{i=1}^{n} x_{ui}x_{v(i+1)}\,.
\end{aligned}$$

A and B are (positive) constants that determine the strength of the constraints. If city i is at position j in the tour, $x_{ij} = 1$. Otherwise $x_{ij} = 0$. If there is a direct edge between nodes u and v in the Hamiltonian cycle, $x_{uv} = 1$. Otherwise, $x_{uv} = 0$.

H_A represents the Hamiltonian cycle of the problem; proper cycles will be at ground state–energy 0. The first two sums ensure each node has exactly one outgoing edge and one incoming edge, respectively. The third sum increases the energy if there is a connection between nodes not in the edge set since it indicates a sub-cycle.

H_B translates the cycle's edge costs to energy. The nested sum adds the weights of all edges in the tour. The ground state will be for the minimum cost tour.

B. Quantum Annealing

To apply the above Hamiltonian in a quantum setting, the binary variables $x_{ij} \in \{0, 1\}$ must be mapped to the Pauli gates using $x_{ij} = \frac{1+\sigma^z_{ij}}{2}$. The coefficients created through this substitution correspond to the vector h (linear terms) and matrix J (quadratic terms).

Once the driver Hamiltonian H_D and problem Hamiltonian H_P are both encoded as the sums of Pauli gates, the system evolves under the time-dependent Hamiltonian $H(t)$:

$$H(t) = A(t)H_D + B(t)H_P\,.$$

The annealing process begins with the system in the ground state of H_D and slowly evolves into H_P. The goal is to remain in the ground state throughout the evolution.

For the TSP, H_P encodes the tour constraints and tour cost, and its ground state corresponds to the minimum-cost tour. Martonak et al. show that quantum annealing can effectively find valid solutions and escape local minima, particularly in small to moderate problem instances [13].

V. Methodology

A. Data Acquisition

For testing data, graphs were downloaded from the online TSPLIB database [17]. Many graphs included hundreds of nodes, so each instance was reduced to a feasible number. Since each graph was complete, this did not change the ability to use the subgraph for the travelling salesman problem. While testing, only graphs that formed metric spaces were used so the results were comparable to Christofides' approximation bound of $\frac{3}{2}$ [4].

B. Data Preparation

First, each graph was converted to an adjacency matrix. To perform quantum annealing on the matrix, we needed to extract the local fields and couplings h and J to use as parameters for the annealing.

We can find the local fields and couplings of the Hamiltonian by expanding the Hamiltonian and simplifying it such that it fits the form seen in Section II-D. First, we expand the sum. Then, we apply the substitution $x_{ij} = \frac{1+\sigma^z_{ij}}{2}$.

$$\begin{aligned}
H_{A_1} &= A\sum_{i=1}^{n}\left(1-\sum_{j=1}^{n}x_{ij}\right)^2 \\
&= A\sum_{i=1}^{n}\left(1-2\sum_{j=1}^{n}x_{ij}+\left(\sum_{j=1}^{n}x_{ij}\right)^2\right) \\
&= A\sum_{i=1}^{n}\left(1-2\sum_{j=1}^{n}x_{ij}+\sum_{j=1}^{n}x_{ij}^2+2\sum_{j<k}^{n}x_{ij}x_{ik}\right) \\
&= A\sum_{i=1}^{n}\left(1-\sum_{j=1}^{n}x_{ij}+2\sum_{j<k}^{n}x_{ij}x_{ik}\right) \\
&= A\sum_{i=1}^{n}1 - A\sum_{i,j=1}^{n}x_{ij} + 2A\sum_{i=1}^{n}\sum_{j<k}^{n}x_{ij}x_{ik} \\
&= -A\sum_{i,j=1}^{n}x_{ij} + 2A\sum_{i=1}^{n}\sum_{j<k}^{n}x_{ij}x_{ik} \\
&= -A\sum_{i,j=1}^{n}\left(\frac{1+\sigma^z_{ij}}{2}\right) + 2A\sum_{i=1}^{n}\sum_{j<k}^{n}\left(\frac{1+\sigma^z_{ij}}{2}\cdot\frac{1+\sigma^z_{ik}}{2}\right) \\
&= -\frac{A}{2}\sum_{i,j=1}^{n}\left(1+\sigma^z_{ij}\right) + \frac{A}{2}\sum_{i=1}^{n}\sum_{j<k}^{n}\left(1+\sigma^z_{ij}+\sigma^z_{ik}+\sigma^z_{ij}\sigma^z_{ik}\right) \\
&= -\frac{A}{2}\sum_{i,j=1}^{n}\sigma^z_{ij} + \frac{A}{2}\sum_{i=1}^{n}\sum_{j<k}^{n}\left(\sigma^z_{ij}+\sigma^z_{ik}+\sigma^z_{ij}\sigma^z_{ik}\right) \\
&= -\sum_{i,j=1}^{n}\frac{A}{2}\sigma^z_{ij} - \sum_{i=1}^{n}\sum_{j<k}^{n} -\frac{A}{2}\left(\sigma^z_{ij}+\sigma^z_{ik}+\sigma^z_{ij}\sigma^z_{ik}\right)\,.
\end{aligned}$$

We can see that the coefficients for the linear terms are $-\frac{A}{2}$ and $\frac{A}{2}$ while the coefficient for the quadratic term is $-\frac{A}{2}$. It is worth noting that since constant terms do not affect the optimization, they were dropped.

The second term of H_A can be expanded similarly to

$$H_{A_2} = -A\sum_{i,j=1}^{n}x_{ij} + 2A\sum_{j=1}^{n}\sum_{i<k}^{n}x_{ij}x_{kj}\,.$$

Performing the same substitution results in

$$H_{A_2} = -\sum_{i,j=1}^{n}\frac{A}{2}\sigma^z_{ij} - \sum_{j=1}^{n}\sum_{i<k}^{n} -\frac{A}{2}\left(\sigma^z_{ij}+\sigma^z_{kj}+\sigma^z_{ij}\sigma^z_{kj}\right)\,.$$

The third term of H_A is dropped since we are only working with complete graphs so the term is always 0.

Transforming H_B gives

$$
\begin{aligned}
H_B &= B \sum_{(uv)\in E} w(uv) \sum_{i=1}^{n} x_{ui}x_{v(i+1)} \\
&= B \sum_{(uv)\in E} w(uv) \sum_{i=1}^{n} \left(\frac{1+\sigma_{ui}^z}{2} \cdot \frac{1+\sigma_{v(i+1)}^z}{2} \right) \\
&= \frac{B}{4} \sum_{(uv)\in E} w(uv) \sum_{i=1}^{n} \left((1+\sigma_{ui}^z) \left(1+\sigma_{v(i+1)}^z \right) \right) \\
&= \frac{B}{4} \sum_{(uv)} w(uv) \sum_{i=1}^{n} \left(1+\sigma_{ui}^z + \sigma_{v(i+1)}^z + \sigma_{ui}^z\sigma_{v(i+1)}^z \right) \\
&= \frac{B}{4} \sum_{(uv)} w(uv) \sum_{i=1}^{n} \left(\sigma_{ui}^z + \sigma_{v(i+1)}^z + \sigma_{ui}^z\sigma_{v(i+1)}^z \right) \\
&= -\sum_{(uv)} \sum_{i=1}^{n} -\frac{B}{4} w(uv) \left(\sigma_{ui}^z + \sigma_{v(i+1)}^z + \sigma_{ui}^z\sigma_{v(i+1)}^z \right) .
\end{aligned}
$$

We can see that the coefficients for the linear and quadratic terms are $-\frac{B}{4}w(uv)$.

C. Testing

Quantum annealing requires n^2 qubits, making it expensive for testing graphs with more than 4-5 nodes. Due to the limitations of currently available quantum hardware and libraries for personal use, tests were performed using OpenJij's *SQASampler* to imitate quantum annealing on a classical computer [14]. This allowed for larger test cases since there were no restrictions on the number of qubits or memory required.

After h and J were extracted from the adjacency matrix, they were converted to Python dictionaries as required by the sampler function. Importantly, the coefficients for h and J needed to have their signs flipped as the simulator followed opposite signing conventions as used when determining the coefficients above.

The sample with the lowest energy is returned from the simulator encoded in Ising spin form. They can then be converted back to binary to decode as a path where $x_{ij} = 1$ if city i is at position j in the tour.

The results of the testing are presented in the next section.

VI. Results

Four graphs were tested, taking subgraphs ranging from 5 to 15 nodes. Each iteration was tested with a pair of parameters: independent annealing attempts (`num_reads`) and annealing lengths (`num_sweeps`).

Figure 1 shows the total number of valid tours across the 4 graphs for each subgraph size and parameter pairing. We can see that as n increases, the simulated quantum annealer does not produce valid paths.

The valid tours for graphs d2103 and pr107 returned by the simulator are plotted against the optimal solution cost in figures 2 and 3. Since there were no valid paths for $n > 9$, the x-axis has been shortened.

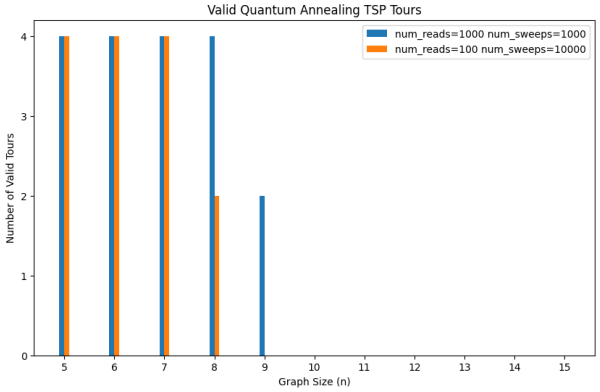

Fig. 1. The number of valid tours produced by the simulator.

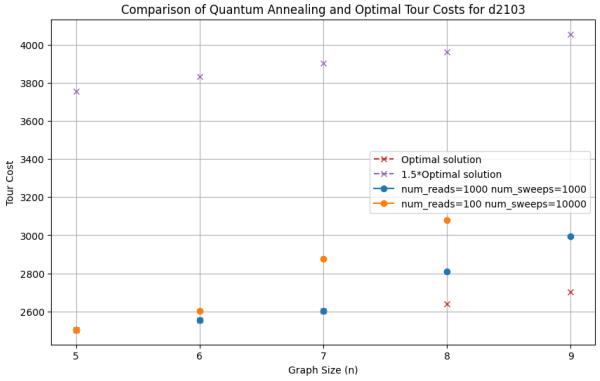

Fig. 2. The tour costs for graph d2103.

The optimal tour costs are found using the Bellman-Held-Karp algorithm sourced from Carl Ekerot's GitHub [5]. As we can see, for small values of n, the simulator produces tour costs very close and sometimes equal to the optimal solution. However, as seen in figure 3 when $n = 9$, the tour cost is greater than 1.5 times the optimal cost, which is the approximation bound of Christofides' algorithm.

Since the quantum annealing simulator only produces valid tours for small values of n and is unbounded even when valid

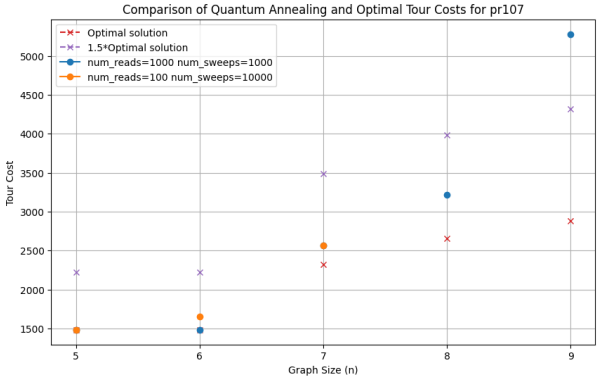

Fig. 3. The tour costs for graph pr107.

tours are found, it does not appear to be a practical algorithm for solving the travelling salesman problem.

As the number of times the program was run consecutively increased along with the annealing time, more valid paths were found, and the costs tended closer to optimal. Considering the results, running the annealer $1\,000$ times with an annealing time of $1\,000$ produces a better result than fewer iterations with a longer annealing time.

As the simulated quantum annealer uses a classical simulator rather than quantum hardware, it is possible that with quantum features such as superposition and teleportation, the annealer could have better success at staying in the ground state.

VII. Discussion

As quantum algorithms have shown the ability to speed up classical algorithms, such as unstructured searching in $O(\sqrt{n})$ using Grover's algorithm [9], there was optimism that other quantum techniques may yield practical improvements for NP-hard problems such as the travelling salesman problem.

Quantum annealing faces significant limitations when applied to large TSP instances. In theory, adiabatic evolution guarantees convergence to the ground state if the system evolves slowly enough. However, for large problems, the gap between the ground and excited states can become extremely small, requiring prohibitively long annealing times to maintain the ground state. In practice, the system is more likely to transition into higher-energy states, leading to suboptimal solutions. As problem size and constraint complexity grow, these effects become more pronounced, limiting the effectiveness of quantum annealing in solving large-scale TSP instances.

Quantum annealing also provides no formal approximation bound, unlike Christofides' classical approximation algorithm, which provides a strict 1.5 approximation bound. Since quantum annealing is heuristic, it only attempts to find minimum-cost tours with high probability but does not guarantee how close the solutions are to the optimal tour or if a valid tour is even found. This puts quantum annealing at a disadvantage compared to classical algorithms that are both efficient and provably near-optimal for well-structured instances, such as metric spaces. Without similar approximation guarantees, quantum annealing remains difficult to justify in settings where reliability and worst-case performance matter.

VIII. Limitations

The implementation of this paper uses a simulated quantum annealer rather than a true quantum annealer. This substitution favoured testing ability since the available quantum libraries could not solve graphs with many nodes. However, a simulation cannot perfectly simulate quantum hardware, so the results do not necessarily indicate how a quantum annealer on quantum hardware would run. There is still skepticism that quantum hardware would introduce significant improvements compared to the simulator since it will always require many qubits and likely need long annealing times.

The simulation took approximately 90 minutes to run for the parameters tested. This made it difficult to test additional parameter pairings such as shorter annealing times with significantly larger rerun counts or very long annealing times with only a few runs.

IX. Future Work

Future work involving quantum annealing and the travelling salesman problem should be the integration of quantum hardware to compare to the simulated quantum annealing. Additional testing could involve alternate parameters or annealing schedules, such as quadratic instead of linear.

Other future work could investigate alternative quantum algorithms. For example, quantum walk-based approaches may provide more scalable or hardware-efficient solutions. Several algorithms already incorporate a form of classical-quantum hybrid frameworks that combine quantum subroutines with classical preprocessing and postprocessing. Perhaps incorporating classical computing during the core processing stage parallel to the quantum processing could introduce more stability.

Another area of future work would be determining the approximation bounds of the algorithm. This would make quantum annealing more comparable to its classical counterparts.

X. Conclusion

This paper examined the use of quantum annealing to solve the travelling salesman problem, comparing its ability to find optimal paths to well-known classical methods.

While quantum annealing offers a strong framework for runtime optimization, it currently suffers from high resource demands and a lack of performance guarantees. As it does not guarantee a valid path and remains unbounded in approximation quality, it is difficult to compare against established classical algorithms with known worst-case performance bounds.

These limitations highlight the gap between conceptual quantum algorithm design and practical implementation in quantum optimization. As such, near-term quantum computing may be better suited to hybrid strategies or problem classes where classical methods fall short.

References

[1] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. draft, 2007.

[2] Richard Bellman. "Dynamic programming treatment of the travelling salesman problem". In: *Journal of the ACM (JACM)* 9.1 (1962), pp. 61–63.

[3] Sergio Boixo et al. "Experimental signature of programmable quantum annealing". In: *Nature communications* 4.1 (2013), p. 2067.

[4] Nicos Christofides. "Worst-case analysis of a new heuristic for the travelling salesman problem". In: *Operations Research Forum*. Vol. 3. 1. Springer. 2022, p. 20.

[5] Carl Ekerot. *held-karp: A pure-Python Held-Karp implementation*. https://github.com/CarlEkerot/held-karp.

[6] Edward Farhi et al. *Quantum Computation by Adiabatic Evolution*. 2000. arXiv: quant - ph / 0001106 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/0001106.

[7] Michael T Goodrich and Roberto Tamassia. *Algorithm design and applications*. Wiley Publishing, 2014.

[8] Sanchit Goyal. "A survey on travelling salesman problem". In: *Midwest instruction and computing symposium*. 2010, pp. 1–9.

[9] Lov K Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 212–219.

[10] Michael Hahsler and Kurt Hornik. "TSP—infrastructure for the traveling salesperson problem". In: *Journal of Statistical Software* 23 (2008), pp. 1–21.

[11] Michael Held and Richard M. Karp. "A dynamic programming approach to sequencing problems". In: *Proceedings of the 1961 16th ACM National Meeting*. ACM '61. New York, NY, USA: Association for Computing Machinery, 1961, pp. 71.201–71.204. ISBN: 9781450373883. DOI: 10.1145/800029.808532. URL: https://doi.org/10.1145/800029.808532.

[12] Andrew Lucas. "Ising formulations of many NP problems". In: *Frontiers in physics* 2 (2014), p. 5.

[13] Roman Martoňák, Giuseppe E Santoro, and Erio Tosatti. "Quantum annealing of the traveling-salesman problem". In: *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 70.5 (2004), p. 057701.

[14] *OpenJij*. https://tutorial.openjij.org/en/intro.html.

[15] Christos H. Papadimitriou. "Computational complexity". In: *Encyclopedia of Computer Science*. GBR: John Wiley and Sons Ltd., 2003, pp. 260–265. ISBN: 0470864125.

[16] Atanu Rajak et al. "Quantum annealing: an overview". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 381.2241 (2022). ISSN: 1471-2962. DOI: 10.1098/rsta.2021.0417. URL: http://dx.doi.org/10.1098/rsta.2021.0417.

[17] *TSPLIB95*. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/index.html.

[18] Sriyani Violina. "Analysis of brute force and branch & bound algorithms to solve the traveling salesperson problem (TSP)". In: *Turkish Journal of Computer and Mathematics Education* 12.8 (2021), pp. 1226–1229.