frontiers in Physics

BRIEF RESEARCH REPORT
published: 12 November 2021
doi: 10.3389/fphy.2021.760783

Solving the Traveling Salesman Problem on the D-Wave Quantum Computer

Siddharth Jain *

Supply Chain Digital & Data Science, Johnson & Johnson, Bridgewater, NJ, United States

The traveling salesman problem is a well-known NP-hard problem in combinatorial optimization. This paper shows how to solve it on an Ising Hamiltonian based quantum annealer by casting it as a quadratic unconstrained binary optimization (QUBO) problem. Results of practical experiments are also presented using D-Wave's 5,000 qubit Advantage 1.1 quantum annealer and the performance is compared to a classical solver. It is found the quantum annealer can only handle a problem size of 8 or less nodes and its performance is subpar compared to the classical solver both in terms of time and accuracy.

OPEN ACCESS

Edited by:
Jingfu Zhang,
Technical University Dortmund, Germany

Reviewed by:
Adrian Paul Flitney,
RMIT University, Australia
Tao Xin,
South University of Science and Technology, China

***Correspondence:**
Siddharth Jain
sjain68@its.jnj.com

Specialty section:
This article was submitted to Quantum Engineering and Technology, a section of the journal Frontiers in Physics

Received: *18 August 2021*
Accepted: *12 October 2021*
Published: *12 November 2021*

Citation:
Jain S (2021) Solving the Traveling Salesman Problem on the D-Wave Quantum Computer. Front. Phys. 9:760783. doi: 10.3389/fphy.2021.760783

Keywords: traveling salesman problem, QUBO, D-wave, quantum annealing, combinatorial optimization

1 INTRODUCTION

The traveling salesman problem is a well-known NP-hard problem in combinatorial optimization supported by an extensive body of research [1]. Given a fully connected graph of n nodes and a $n \times n$ adjacency matrix M whose entries give the cost to travel from node i to node j and $M_{ii} = 0$, the goal is to output a roundtrip (cycle) that minimizes the cost of traversing all the nodes in the graph (each node is traversed exactly once and so has exactly one ingress and one egress). We restrict ourselves to the case when the cost to travel from node i to node j is same as the cost to travel from node j to node i (negative values of cost are permissible) and is further given by $M_{ij} + M_{ji}$. If we let X be a $n \times n$ symmetrical indicator matrix consisting of 0 s and 1 s ($x_{ij} = 1$ if node i is adjacent to node j in the output cycle; we will also call this as the nodes being connected in the output cycle or graph; this is not to be confused with whether the nodes are connected in the input graph which they always are), we seek to find the X that minimizes $\sum M. * X$ where $.*$ denotes element-wise multiplication of matrices and $\sum$ denotes sum over elements of the matrix.

This paper describes a quantum solution to this problem that leverages adiabatic quantum computation. Adiabatic quantum computation is one of the two paradigms of quantum computation—gate-based model being the other one. In adiabatic quantum computation, a system of qubits is prepared in an initial ground state associated with the Hamiltonian H_0 and the system is gradually evolved to a target state whose Hamiltonian is given by H_1. If the time evolution is done slowly enough, the theorem of adiabatic quantum computation guarantees that the qubits will track the ground state of the evolving Hamiltonian and at the end they will be found in the ground state corresponding to H_1. Mathematically, if

$$H(t) = (1 - t)H_0 + tH_1 \tag{1}$$

where t is some normalized unit of time, then $H(0) = H_0$, $H(1) = H_1$. H_0 is the starting Hamiltonian (initial conditions) and at time $t = 1$ the qubits will assume values that minimize the energy associated with H_1.

Up until now the discussion is general. D-Wave's[1] quantum computers are special purpose devices that aim to realize adiabatic quantum computation through quantum annealing where the Hamiltonian is constrained to be an Ising Hamiltonian. The energy (eigenvalues) associated with the target state H_1 is given by:

$$E_{Ising} = -\sum_i h_i s_i - \sum_i \sum_{j,j\neq i} J_{ij} s_i s_j \tag{2}$$

The Ising constraint is a natural consequence of how the device is built and wired. It is a system of superconducting qubits obeying the Ising model. In **Eq. 2**, h_i are biases and J_{ij} are coupling constants. s_i are the spins of the qubits $s_i \in \{-1, +1\}$. The H_0 state is realized by subjecting the system to a high transverse magnetic field in the x direction ($H_0 = X \otimes \ldots \otimes X$ where X is Pauli spin matrix along x) which aligns all the spins in the x direction and thus sets the ground state to an equal superposition of all states in the Z basis and the external magnetic field is gradually turned off to attain the state corresponding to H_1. The spins will automatically align themselves so as to minimize the energy given by **Eq. 2**. This process is known as quantum annealing.

A simple change of variables ($x = (1 + s)/2$) together with the observation that $x^2 = x$ when $x \in \{0, 1\}$ transforms **Eq. 2** into (ignoring constant term as it is of no consequence)

$$E = x^T Q x \tag{3}$$

where Q is a Hermitian matrix (symmetric if dealing with real numbers only). The minimization of **Eq. 3** is a Quadratic Unconstrained Binary Optimization (QUBO) problem. D-Wave's devices are tailored to solve this class of problems.

2 QUADRATIC UNCONSTRAINED BINARY OPTIMIZATION FORMULATION OF THE TRAVELING SALESMAN PROBLEM

The objective function of the traveling salesman problem can then be written as:

$$f = \sum M. * X = \sum_{i=0}^{i<n} \sum_{j=i+1}^{j<n} \left(M_{ij} + M_{ji}\right) x_{ij} = \sum_{k=0}^{k<m} B_k b_k \tag{4}$$

where $M_{ij} + M_{ji}$ is the cost of traveling from i to j (or vice-versa) and we have used indices starting from 0 which are convenient for turning equations into computer code. The x_{ij} variables are mapped to b_k and m is given by ${}^nC_2 = n\ (n-1)/2$. Thus for a problem of size n, we have to solve for $m = {}^nC_2$ unknowns. f is a linear function in b_k and the solution is trivial if there are no constraints on b_k (set $b_k = 1$ if $B_k < 0$ and 0 otherwise). However, there are constraints on what b_k can be. Each node is traversed only once and connects to only two other nodes in the output cycle. These constraints can be written as n equations (one equation for each i):

$$\forall i \in \{0, \ldots, n-1\} \ \sum_j x_{ij} = 2 \tag{5}$$

These constraints have to be mapped to b_k (as we solve for b_k ultimately) and the function mapping i, j indices to k (and thus x_{ij} to b_k) is given by:

$$k(i, j, n) = \begin{cases} \text{undefined} & \text{for } i = j \\ k(j, i, n) & \text{for } i > j \\ in - i(i+1)/2 + j - (i+1) & \text{for } i < j \end{cases} \tag{6}$$

So far what we have is a linear integer programming problem subject to linear constraints. To convert it into a QUBO, we formulate each constraint as:

$$\forall i \in \{0, \ldots, n-1\} \ \min\left(2 - \sum_j x_{ij}\right)^2 \tag{7}$$

and use the method of Lagrange multipliers to absorb the constraints into the objective function as follows:

$$q = \sum_{k=0}^{k<m} B_k b_k + \sum_{i=0}^{i<n} \lambda_i \left(2 - \sum_{j,j\neq i} b_{k(i,j)}\right)^2 \tag{8}$$

where λ_i are Lagrange multipliers. The minimization of **Eq. 8** is a QUBO problem. The Ising (and equivalently QUBO) formulation of many NP problems can be found in [2]. The QUBO class of problems is NP-hard.

3 IMPLEMENTATION

We have used D-Wave's 5,436 qubit Advantage 1.1 QPU [3] to solve instances of **Eq. 8**. We created 8 sample problems all of size $n = 8$ (corresponding to $m = 28$ binary variables) in which the entries of M were randomly sampled from $[1, n]$ or $\{1, 2, \ldots, 8\}$. All the Lagrange multipliers (λ_i) were set equal to each other and further equal to the maximum of the absolute values in the matrix M (thus 8 in this case). The Lagrange multiplier acts as a weight given to the constraint. It should be set high enough to ensure the constraint is satisfied, but setting it too high obscures the real function we are trying to minimize. In addition to the Lagrange multipliers, there are some other parameters that can affect the solutions returned by the QPU such as the chain strength and annealing time. We use the default annealing time of 20 μs of the QPU. The chain strength parameter is related to minor embedding of the problem graph into the actual physical connectivity graph of the QPU. The qubits represented by the variables b_k are logical qubits of the problem and we assume in our problem formulation that any of them can interact with another. However, in the physical processor the qubits are laid out in a certain fixed pattern (graph) and to emulate the problem graph it has to be minor embedded into the physical graph. This is a hard problem in its own right (see [4] and references cited therein) and in general given m QUBO

[1]https://www.dwavesys.com/

variables, we require $O(m^2)$ physical qubits for minor embedding. That's why even though the processor has 5,436 qubits, its compute power translates to the order of $\sqrt{5436} \approx 73$ logical qubits. The task of minor embedding the problem graph into the QPU is akin to compilation of a classical computer program to the instruction set of the CPU. We use the default function provided by D-Wave (`minorminer.find_embedding`)[2] to calculate the minor embedding of our problem graph. Once, the problem graph is embedded into the physical graph, a chain of physical qubits maps to a logical qubit. Long chains should be avoided and can create problems when the constituent qubits of a chain assume different values - this is known as a chain break and can result in garbage solutions. The chain strength parameter is a parameter that avoids the chain from breaking. Like the Lagrange multiplier, it should be set high enough but setting it too high will hide the actual problem we want to solve. In our implementation we use the `scaled`[3] function provided by D-Wave to automatically set the chain strength without any manual ad-hoc fine tuning. With this, we found the auto-computed chain strength to be between 30.5 and 32 for all the problem cases and there were no chain breaks in the returned solutions. If the annealing process were perfect, the wavefunction would be in the ground state of target Hamiltonian in the end, but in practice that is not always the case. Therefore we perform multiple runs (samples)—100 runs for each problem in our case. We then iterate through the samples (observations) in order of lowest energy and select the first sample that satisfies the constraints of **Eq. 5**. This marks the solution to the problem by the quantum method.

To compare the results and performance of the quantum solution, we also solved each of the problem cases by trying out all possible combinations (brute-force). For a problem of size n, the total number of possible combinations is given by $(n-1)!/2$. For $n = 8$ this translates to 2,520. **Figure 1** shows graph of the original objective function $\sum M.*X$ against all the combinations for a sample problem. It has many local extrema and no recognizable structure. All the 8 problems are characterized by closely spaced extrema - the difference in the objective function between the best and second-best input configuration ranges from 1–3. This equals the energy difference δE between the ground and first-excited state of the quantum system. Quantum annealing performs best when this energy gap is high. When the energy gap is small, the system can jump to a excited state over course of the time evolution giving suboptimal solution in the end.

A classical optimizer for the traveling salesman problem was also developed. It uses the following simple heuristic: we start with a random initial configuration (cycle). Then, we pick 2 nodes at random and check if swapping their positions (order) would lead to an improvement of the objective function. If so, we swap and repeat the process. If we have tried all ${}^nC_2$ swaps and got no improvement of the objective function, the algorithm terminates. This algorithm is guaranteed to converge (and converges fast) but it does not always converge to the global optimum. The chances of converging to the global optimum decline with increase in size of the problem n.

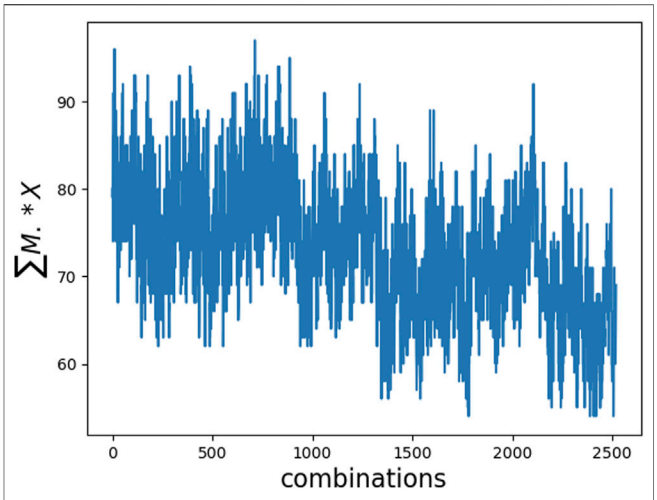

FIGURE 1 | Graph of objective function against all possible combinations for a sample problem.

4 RESULTS

Table 1 shows our results and compares the performance of all the 3 solvers. Overall we find the quantum solver performed worse than the classical solver. In terms of time too, the classical solver ran faster for $n = 8$. The quantum solver takes 20 µs × 100 samples = 2 ms to run and this time excludes the time spent in minor embedding of the problem graph. In comparison the classical solver implemented in Python took less than a ms to run for each of the problems on a 2018 Macbook Pro equipped with a 2.2 GHz Intel Core i7 processor and 16 GB 2400 MHz DDR4 RAM. $n = 8$ is also the maximum size of the problem the quantum solver was able to handle. When we tried problems with $n = 9$ ($m = 36$), the D-Wave QPU did not return any valid solution for 5 out of 8 test cases with the methods described in previous section, and for $n = 10$ ($m = 45$), the QPU returned a solution for only 1 out of 8 test cases. The solutions were also suboptimal compared to classical solver. They are not listed in the main paper but can be found in the **Supplementary material** accompanying the paper. One strength of our implementation is that no tweaks are needed to the code and the same code that works when entries of M are sampled from $[1, n]$ also works as-is when the entries are sampled from $[-a, +a]$.

[2]https://github.com/dwavesystems/minorminer/blob/main/minorminer/minorminer.py

[3]https://github.com/dwavesystems/dwave-system/blob/master/dwave/embedding/chain_strength.py

TABLE 1 | Evaluation of quantum algorithm and its comparison to a classical solver as well as brute-force method (n = 8).

	Brute force			Quantum solver				Classical solver			
	Obj.	Qubo energy	# Of distinct solutions	Obj.	Qubo energy	Is optimal	Is second best	Obj.	# Of steps	Is optimal	Is second best
Problem 1	54	−202	5	61	−195	×	×	54	77	✓	
Problem 2	54	−202	1	68	−188	×	×	54	62	✓	
Problem 3	41	−215	1	53	−203	×	×	44	60	×	✓
Problem 4	49	−207	2	55	−201	×	×	49	94	✓	
Problem 5	40	−216	1	54	−202	×	×	40	70	✓	
Problem 6	52	−204	2	58	−198	×	×	52	85	✓	
Problem 7	51	−205	1	63	−193	×	×	59	29	×	×
Problem 8	47	−209	1	59	−197	×	×	50	60	×	×

Obj. is the original objective function $\sum M.*X$. *The QUBO energy is same as* ***Eq. 8*** *minus the constant term in it. The number of steps is equal to the number of combinations tried by the classical solver. It is not equal to the number of swaps performed.*

One advantage of the classical solver is that it always respects the constraints of the problem (**Eq. 5**) whereas the quantum solution turns the constraints into soft-constraints that can be violated and have to be checked for satisfaction. Another advantage is that the quantum solver can return incorrect solutions in which there is more than 1 loop (cycle) e.g., it might configure 4 nodes to be in one loop and 4 others to be in another loop—there is no constraint in **Eq. 5** preventing that. Each node is still connected to two other nodes. The classical solver does not suffer from this limitation.

For reference, D-Wave also provides a code sample to solve the TSP using a QPU.[4] We tried that code as well but it resulted in errors. The formulation provided here is better than D-Wave's solution because it takes into account that $x_{ij} = x_{ji}$ (consequently there are ${}^{n}C_{2}$ variables to solve for) whereas in D-Wave's solution this constraint is treated as a soft-constraint using Lagrange multipliers and the total number of variables is $n(n-1)$. The classical optimizer we have described is the 2-opt algorithm for solving the TSP [5]. Several computer codes and packages for classical solutions to TSP can be found online.[5, 6, 7] A quantum study using D-Wave to solve the related Capacitated Vehicle Routing Problem (CVRP) can be found in [6].

Lastly, in this paper we restricted ourselves to the case when the cost of traveling from i to j is same as the cost of traveling from j to i. The restriction can be removed and the methods developed here can be applied to the general case as well. See the **Appendix** for details.

In conclusion, this paper presented a QUBO formulation of the traveling salesman problem which makes it amenable to quantum annealers and although our experiments with D-Wave QPU did not demonstrate any quantum advantage, we remain hopeful that with advancements in technology the quantum solution might outperform classical method esp. with large n although quantum annealers like the D-Wave QPU will find a challenge since the number of qubits required grows as m^2 due to the minor embedding problem. Hence, simply adding more qubits to the QPU while keeping the rest of the architecture and design the same is unlikely to succeed. Better innovations will be required.

DATA AVAILABILITY STATEMENT

All computer code and data is attached as **Supplementary Material** and licensed under terms of Apache 2.0 License[8].

AUTHOR CONTRIBUTIONS

SJ conceived the problem and its solution, wrote computer code, performed all experiments, analyzed data and wrote the paper.

FUNDING

This work was funded by Supply Chain Digital & Data Science, Johnson & Johnson. Johnshon & Johnson had no role in study design, collection, analysis, interpretation of data, or the writing of this article.

DEDICATION

I dedicate this paper to Pink Floyd — The Endless River.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fphy.2021.760783/full#supplementary-material

[4]https://github.com/dwavesystems/dwave-networkx/blob/main/dwave_networkx/algorithms/tsp.py

[5]https://developers.google.com/optimization/routing/tsp

[6]https://python-mip.readthedocs.io/en/latest/examples.html

[7]https://www.math.uwaterloo.ca/tsp/concorde.html

[8]https://www.apache.org/licenses/LICENSE-2.0.txt

REFERENCES

1. Applegate DL, Bixby RE, Chvátal V, and Cook WJ. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press (2007).
2. Lucas A. Ising Formulations of many NP Problems. *Front Phys* (2014) 2:5. doi:10.3389/fphy.2014.00005
3. McGeoch C, and Farre P. The D-Wave Advantage System: An Overview. Technical Report. Burnaby, BC (2020). 14-1049A-A.
4. Date P, Patton R, Schuman C, and Potok T. Efficiently Embedding QUBO Problems on Adiabatic Quantum Computers. *Quan Inf Process* (2019) 18:1–31. doi:10.1007/s11128-019-2236-3
5. Croes GA. A Method for Solving Traveling-Salesman Problems. *Operations Res* (1958) 6:791–812. doi:10.1287/opre.6.6.791
6. Feld S, Roch C, Gabor T, Seidel C, Neukart F, Galter I, et al. A Hybrid Solution Method for the Capacitated Vehicle Routing Problem Using a Quantum Annealer. *Front ICT* (2019) 6:13. doi:10.3389/fict.2019.00013

Conflict of Interest: Author SJ was employed by the company Johnson & Johnson.

APPENDIX

This appendix describes QUBO formulation of the TSP when the cost of traveling from node i to node j is not equal to the cost of traveling from node j to node i. In this case the total number of possible combinations is given by $(n-1)!$ instead of $(n-1)!/2$ and for a problem of size n, the number of binary variables to solve for becomes $2 \times {}^{n}C_2$ instead of ${}^{n}C_2$. This is also equal to the number of entries in the matrix X minus the diagonal. Let x_{ij} be the binary variable that denotes if there is a traversal from node i to node j. The objective function $\sum M.*X$ remains unchanged but X is no longer a symmetric matrix. Each node i must have exactly one ingress which is represented by n constraints:

$$\forall i \sum_{j} x_{ji} = 1 \tag{9}$$

The above constraint can be equivalently stated as each column of matrix X should have exactly one entry with a 1 in it. Similarly each node must have exactly one egress which is represented by another set of n constraints (equivalently stated as each row of matrix X should have exactly one entry with a 1 in it):

$$\forall i \sum_{j} x_{ij} = 1 \tag{10}$$

Finally, if x_{ij} is 1 then $x_{ji} = 0$ (X is not a symmetric matrix). This gives following ${}^{n}C_2$ constraints:

$$\forall (i, j)\ x_{ij} + x_{ji} \leq 1 \tag{11}$$

We have seen before how to absorb the linear constraints into the cost function using Lagrange multipliers. Each inequality in **Eq. 11** can be represented by corresponding penalty function $x_{ij}x_{ji}$ which is 0 if the inequality is satisfied and 1 otherwise. We can capture all the constraints into following QUBO cost function:

$$\begin{aligned} q = &\sum M.*X \\ &+ \lambda_1 \sum_{i=0}^{i<n} \left(1 - \sum_{j,j\neq i} x_{ij}\right)^2 \\ &+ \lambda_2 \sum_{i=0}^{i<n} \left(1 - \sum_{j,j\neq i} x_{ji}\right)^2 \\ &+ \lambda_3 \sum_{i=0}^{i<n} \sum_{j,j\neq i} x_{ij}x_{ji} \end{aligned} \tag{12}$$

where λ_i are Lagrange multipliers. Minimization of above will solve the TSP for the extended case.