A Review on Quadratic Unconstrained Binary Optimization, Ising model and Quantum Annealing

Soumik Samanta*[1], Ayan Barui[2]

[1]Department of Physics, St. Paul's Cathedral Mission College, University of Calcutta, Kolkata, India

[2]Department of Physics, St. Paul's Cathedral Mission College, University of Calcutta, Kolkata, India

July 28, 2022

Abstract

The Quadratic Unconstrained Binary Optimization (QUBO) is well-known class of Combinatorial optimization problems with applications ranging from economics and computer science to pure mathematics. In this report, we explored the connection and equivalence of these QUBO problems with certain mathematical model called the Ising model, named after Ernst Ising is used in statistical mechanics to describe ferromagnetism. As QUBO problems mainly belong to the NP-Hard category in terms of computational complexity, getting exact solutions with reasonable efficiency becomes a hefty task for classical computers. This is where the connection to the Ising model becomes fruitful as it opens a bridge to encode QUBOs into physical systems capable of yielding Heuristic solutions to the problem. We also explore how Adiabatic Quantum Computing and Quantum Annealing are used to develop heuristic solutions and the difference these systems have from the classical approach. At the end of the report, we have simulated a set of three problems - Maximum Cut, Minimum Vertex Cover and Travelling Salesman Problem and solved them with the help of D-Wave systems' Quantum Annealers.

Keywords: Quadratic Unconstrained Binary Optimization (QUBO), Ising model, Quantum Annealing, NISQ Algorithms

*soumiksamanta10@gmail.com

Contents

1 QUBO - Quadratic Unconstrained Binary Optimization **3**
1.1 QUBO . . . 3
1.1.1 Max-cut problem . . . 4
1.2 Constrained to unconstrained re-formulation . . . 6
1.2.1 Minimum vertex cover problem . . . 7

2 Ising Model **9**
2.1 Ising model . . . 9
2.2 Connection between Ising model and QUBO . . . 10

3 Quantum Annealing and D-wave Systems **11**
3.1 Simulated Annealing . . . 11
3.2 Adiabatic Quantum Computing and Quantum Annealing . . . 11
3.3 Quantum Annealing for Ising model/QUBO problems . . . 13
3.4 Chimera Topology in D-Wave systems . . . 14
3.5 Minor Embedding . . . 15

4 Simulation results **17**
4.1 Simple QUBO . . . 17
4.2 The Maximum cut problem . . . 18
4.3 Minimum vertex cover problem . . . 19
4.4 Travelling salesman problem . . . 21

Chapter 1

QUBO - Quadratic Unconstrained Binary Optimization

1.1 QUBO

QUBO (Quadratic Unconstrained Binary Optimization) is a mathematical formulation used to solve various Combinatorial Optimization (CO) problems. Combinatorial Optimization (CO) is one of the essential areas of optimization with practical applications found in multiple industries. It involves optimizing an objective function, given a finite collection of objects and a set of constraints. Simply put, CO problems are concerned with making wise choices in settings where a large number of yes/no decisions are to be made, with each set of decisions yielding a corresponding objective function value. As the number of objects and constraints increases, finding an optimal solution becomes increasingly difficult. The traditional approach of developing an algorithm tailored to a specific problem is inefficient as it requires the creation of a diverse array of solution techniques, with each having limited application outside of its own problem.

It has been recently discovered that QUBO can be used to solve an exceptional range of important CO problems found in industry, as documented in [4] and [5]. Combinatorial problems can be modelled into a QUBO problem. Once given a QUBO formulation, these problems can be solved to yield solutions whose quality in many cases rivals or even surpasses the quality of the solutions produced by the best specialized methods.

Definition 1.1.1 *The quadratic unconstrained binary optimization problem is defined as*

$$\begin{aligned} &min \quad y = x^t Q x \\ &such\ that \quad x \in S \end{aligned} \tag{1.1}$$

where S represents the binary discrete set $\{0, 1\}$ or $\{-1, 1\}$ and Q is a $n \times n$ symmetric or upper triangular matrix. This can be achieved without loss of generality simply as follows:

Symmetric form: For all i and j except $i = j$, replace q_{ij} by $(q_{ij} + q_{ji})/2$
Upper triangular form: For all i and j with $j > i$, replace q_{ij} by $q_{ij} + q_{ji}$. Then replace all q_{ij} for $j < i$ by 0.

This simple modelling of combinatorial optimization as a QUBO has many applications. For example, the use of these models for representing and solving optimization problems on graphs, facility locations, resources allocation, clustering, set partitioning, sequencing and ordering, various forms of assignment problems and many others have been reported in the literature.

Example 1 *Consider the optimization problem:*

$$Min \quad y = -6x_1 - 2x_2 - 7x_3 - 6x_4 + 3x_1x_2 + 7x_1x_3 + 2x_2x_3 + 11x_3x_4$$

Where the variables x_j, are binary. We can make several observations:

1. *Since, the order of the function to be minimized is 2, the function is quadratic in binary variables with linear part $-6x_1 - 2x_2 - 7x_3 - 6x_4$ and a quadratic part $3x_1x_2 + 7x_1x_3 + 2x_2x_3 + 11x_3x_4$*

2. *Since binary variables satisfy $x_j = x_j^2$, the linear part can be written as $-6x_1^2 - 2x_2^2 - 7x_3^2 - 6x_4^2$*

3. *Then we can rewrite the function as $y = 6x_1^2 - 2x_2^2 - 7x_3^2 - 6x_4^2 + 3x_1x_2 + 7x_1x_3 + 2x_2x_3 + 11x_3x_4$.*

$$y = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix} \begin{bmatrix} -6 & 3 & 7 & 0 \\ 3 & -2 & 2 & 0 \\ 7 & 2 & -7 & 11 \\ 0 & 0 & 11 & -6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

4. *This can be written in the matrix notation as in definition,*

$$Minimize\ y = x^t Qx \tag{1.2}$$

   where x is a column vector of binary variables.

5. *Notice that the coefficients of the original linear terms appear on the main diagonal of Q matrix and in this case Q is symmetric without needing to modify the values as per the definitions.*

6. *QUBO is an unconstrained model with all problem data being contained in the Q matrix. These characteristics make the QUBO model particularly attractive as a modelling framework for combinatorial optimization problems, offering a novel alternative to classically constrained representations.*

1.1.1 Max-cut problem

The max-cut problem is a well-known combinatorial optimization problem which aims to find a partition of the vertex set of a graph $G(V, E)$ into two parts such that the sum of weights over all the edges connecting two vertex subsets in a given weighted unidirectional graph is maximum. For an unweighted graph, weights over all the edges are equal to 1. Max-cut problems are difficult to

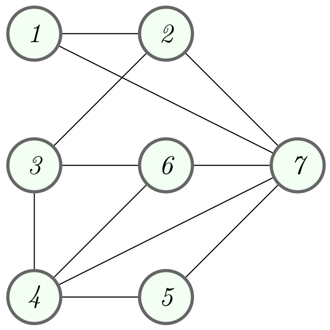
solve; it was proved to be NP-hard problems[2]. We can model this problem by introducing binary variables satisfying $x_j = 1$ if vertex j is in one set and $x_j = 0$ if it is in the other set. Viewing a cut as severing edges joining two sets to leave endpoints of the edges in different vertex sets, the quantity $x_i + x_i - 2x_i x_j$ identifies whether the edge (i, j) is in the cut or not. Notice that if $x_i + x_i - 2x_i x_j = 1$ then exactly one of x_i or x_j equals 1. Otherwise, $x_i + x_i - 2x_i x_j = 0$ and the edge (i, j) is not in the cut.
Thus maximizing the number of edges in cut is modeled as

$$\text{maximize } z = \sum_{(i,j)\in E} (x_i + x_j - 2x_i x_j) \tag{1.3}$$

we can model the maximization as minimization of negative of objective function

$$\text{minimize } y = -z = \sum_{(i,j)\in E} (2x_i x_j - x_i - x_j) \tag{1.4}$$

Example 2 *Consider the following undirected graph with 7 vertices and 11 edges*



Explicitly taking all the edges in graph gives the following formulation:

$$\begin{aligned} \textit{maximize } z = {} & (x_1 + x_2 - 2x_1x_2) + (x_2 + x_3 - 2x_2x_3) + (x_1 + x_7 - 2x_1x_7) \\ & +(x_2 + x_7 - 2x_2x_7) + (x_3 + x_6 - 2x_3x_7) + (x_3 + x_4 - 2x_3x_4) \\ & +(x_4 + x_6 - 2x_4x_6) + (x_4 + x_5 - 2x_4x_5) + (x_4 + x_7 - 2x_4x_7) \\ & +(x_5 + x_7 - 2x_5x_7) + (x_6 + x_7 - 2x_6x_7) \end{aligned} \tag{1.5}$$

or

$$\begin{aligned} \textit{min } y = {} & -2x_1 - 3x_2 - 3x_3 - 4x_4 - 2x_5 - 3x_6 - 5x_7 + 2x_1x_2 + 2x_2x_3 + 2x_1x_7 \\ & +2x_2x_7 + 2x_3x_6 + 2x_3x_4 + 2x_4x_6 + 2x_4x_5 + 2x_4x_7 + 2x_5x_7 + 2x_6x_7 \end{aligned} \tag{1.6}$$

which is equivalent to

$$min\ y = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{bmatrix} \begin{bmatrix} -2 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & -3 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & -3 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & -4 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & -2 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & -3 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & -5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} \tag{1.7}$$

which is in the QUBO model form

$$min\ y = x^t Q x$$

But, most real-world problems contain constraints that must be satisfied as the optimizer searches for a good solution. Constrained binary optimization problems can be converted into unconstrained optimization problems by a reformulation procedure. The reformulation procedure re-casts a constrained problem into an equivalent unconstrained binary model.

1.2 Constrained to unconstrained re-formulation

Reformulations are accomplished by including quadratic infeasibility penalties in the objective function as an alternative to explicitly imposing constraints. Any linear or quadratic problem with linear constraints and bounded integer variables can, in principle, be reformulated as QUBO using quadratic penalties.

The penalties introduced are chosen such that the original constraints (problem) are satisfied by the natural functioning of the solver as it looks for solutions that avoid incurring the penalties. That is, the penalties are formulated so that they equal zero for feasible solutions and equal some positive penalty amount for infeasible solutions. For many simple constraints, appropriate quadratic penalties are known in advance. Some of such penalties are shown in Table 1.1, where P is a large positive scalar.

Classical Constraint	Equivalent Penalty
x+y ≤ 1	P(xy)
x+y ≥ 1	P(1-x-y+xy)
x+y=1	P(1 - x - y+2xy)
x $\leq y$	P(x-xy)
x$_1 + x_2 + x_3 \leq 1$	P$(x_1x_2 + x_1x_3 + x_2x_3)$
x=y	P(x+y-2xy)

Table 1.1: Known penalties

Note that the penalty term in Table 1.1 corresponding to classical constraints is zero if the associated constraint is satisfied, and otherwise the penalty is positive. These penalties, then, can be used as an alternative to explicitly introducing the original constraints. However, for some constraints, penalty functions are not known in advance, and we need to find an appropriate penalty term. A simple procedure for finding the penalty for any linear constraint is given as follows:

Consider the general constrained problem of form

$$\begin{aligned} \min\ y &= x^t Q x \\ \text{s.t } Ax &= b \\ x &\text{ is binary} \end{aligned} \tag{1.8}$$

If the constraints are inequalities, we can convert them into equalities by introducing slack or surplus variables. These constrained quadratic optimization models are converted into equivalent QUBO models by adding a quadratic infeasibility penalty function to the objective function in place of explicitly imposing the constraints $Ax = b$.

Specifically, for a positive scalar P:

$$\begin{aligned} y &= x^t Q x + P(Ax-b)^t(Ax-b) \\ &= x^t Q x + x^t D x + c \\ &= x^t \hat{Q} x + c \end{aligned}$$

where the matrix D and the additive constant c result directly from the matrix multiplication indicated. Dropping the additive constant, the equivalent unconstrained version of the constrained problem 1.8 becomes

$$\begin{aligned} \min\ y &= x^t \hat{Q} x \\ x &\text{ is binary} \end{aligned} \tag{1.9}$$

1.2.1 Minimum vertex cover problem

Vertex cover problem(VCP) is defined over an undirected graph $G(V, E)$ and searches for a subset S of the vertex set V such that for each edge in E, at least one of its endpoints is in S. In minimum vertex cover problem, the cardinality of S, i.e. $|S|$ is as minimum as possible.

The Minimum vertex cover(MVC) problem is formulated as follows:

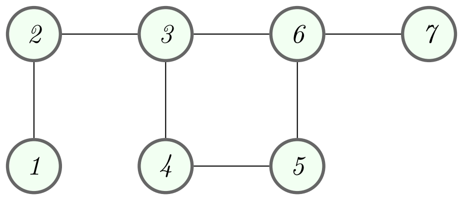
$$\begin{aligned} \min\ y &= \sum_{i \in V} c_i x_i \\ \text{s.t } x_i + x_j \geqslant 1&, \quad \forall (i,j) \in E \\ x &\text{ is binary} \end{aligned} \tag{1.10}$$

Here c_i is the cost(or weight) associated with vertices. From 1.1, the constraints in MVC problem can be represented by $P(1-x-y+xy)$. Therefore, the unconstrained alternative to the constrained MVC problem is

$$\min\ y = \sum_{i \in V} c_i x_i + P \sum_{(i,j) \in E} (1 - x_i - x_j + x_i x_j) \tag{1.11}$$

where P is positive scalar penalty. we can write 1.11 as min $x^t Q x$ plus a constant term.

Example 3 *Consider the following graph with 7 vertices and 7 edges*



MVC is formulated as

$$\begin{aligned} min\ y = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \\ s.t\ x_1 + x_2 \geqslant 1 \\ x_2 + x_3 \geqslant 1 \\ x_3 + x_4 \geqslant 1 \\ x_4 + x_5 \geqslant 1 \\ x_5 + x_6 \geqslant 1 \\ x_3 + x_6 \geqslant 1 \\ x_6 + x_7 \geqslant 1 \end{aligned} \tag{1.12}$$

equivalent unconstrained model of 1.12 is

$$\begin{aligned} min\ y = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + P(7 - x_1 - 2x_2 - 3x_3 - 2x_4 \\ -2x_5 - 3x_6 - x_7 + x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6 + x_3x_6 + x_6x_7) \end{aligned} \tag{1.13}$$

Arbitrarily choosing P to be 2 and droping constant $(7P = 14)$ we get QUBO model min x^tQx with Q matrix is given by

$$\begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -5 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & -3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -3 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -5 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

Solving the QUBO model gives a minimum value $x^tQx = -11$ which gives $y = 14 - 11 = 3$ at $x = (0, 1, 0, 1, 0, 1, 0)$, meaning that a minimum cover is given by $S = 2, 4, 6$ vertex set.It is easy to verify that all penalty terms are 0 at this solution.

Note - instead of taking equal scalar penalty values, we can choose different penalty scalars also. If a constraint must absolutely be satisfied, i.e., a “hard” constraint, then P must be large enough to preclude a violation. Some constraints, however, are “soft”, meaning it is desirable to satisfy them but slight violations can be tolerated. For such cases, a more moderate penalty value will suffice. Too large penalty scalar delays the solution, making it difficult to distinguish the quality of one solution from another. On the other hand, too small a penalty jeopardizes the exploration of feasible solutions.

Chapter 2

Ising Model

2.1 Ising model

The Ising model is a simple model of ferromagnetism that provides some insight into how phase transitions and the non-analytic behaviour of thermodynamic quantities across phase transitions occur in physics. This model can be described as a collection of a finite number of lattice sites with up/down spin(scalar fields), i.e., $+1$ or -1 at each lattice point.
The two assumptions in this mathematical model are:

- The spins can only be in two states, i.e., UP or DOWN.
- The interaction between only neighbouring(or adjacent) spins is considered while calculating the Hamiltonian and the interaction strength is same.

The Hamiltonian of this configuration is then given as:

$$H(s_1, s_2, ..., s_N) = -\sum_{i<j} J_{ij} s_i s_j - \sum_{i=1}^{N} h_i s_i \tag{2.1}$$

where,

J_{ij}	$\rightarrow$	interaction term, which depends upon coupling strength of adjacent magnetic moments.
s_i, s_j	$\rightarrow$	adjacent spins
h_i	$\rightarrow$	external field

The quantum version of this Hamiltonian is:

$$H_P = H(\sigma_1^z, ..., \sigma_N^z) \tag{2.2}$$

where σ_i^z is a Pauli matrix acting on the i^{th} qubit in a Hilbert space of N qubits. The system has a tendency to assume the lowest possible energy state, and the neighbouring spin interaction alters the spin to assume the lowest possible energy state.
In QUBO problems, we are optimizing unconstrained binary problems. Since Ising models only have a discrete binary variable, we can use the idea of minimising a QUBO problem to find out the Hamiltonian.

2.2 Connection between Ising model and QUBO

Mathematically, a problem is said to be NP-complete if we can find a mapping to the decision form of the Ising model with a polynomial number of steps. This mapping can be re-interpreted as a pseudo-Boolean optimization problem [1]. It can be proved that the Ising model is mathematically equivalent to a QUBO problem.

As we saw from equation [2.1], the basic Hamiltonian of the Ising model depends on the discrete set of spins s_i, s_j, which only takes values from {+1,-1}. On the same footing, we have QUBO model where binary decision variables x_i, x_j takes values belonging to only {0,1}. So, we can relate spin variables of the QUBO model with the linear relation;

$$x_i = \frac{1+s_i}{2} \qquad (\because s_i = 2x_i - 1) \tag{2.3}$$

From this relation, we will have:

$$\begin{array}{rcl} x_i = 1 & \text{if} & s_i = 1 \\ x_i = -1 & \text{if} & s_i = 0 \end{array}$$

So, now substituting the value of s_i from equation [2.3] in the Hamiltonian of Ising model (equation [2.1]), we get:

$$\begin{aligned} H(x_1, x_2, ..., x_N) &= -\sum_{i<j} J_{ij}(2x_i - 1)(2x_j - 1) - \sum_{i=1}^{N} h_i(2x_i - 1) \\ &= -\sum_{i<j} 4J_{ij}x_ix_j + \sum_{i<j} 2J_{ij}x_i + \sum_{i<j} 2J_{ij}x_j - \sum_{i<j} 4J_{ij} - \sum_{i=1}^{N} 2h_ix_i + \sum_{i=1}^{N} h_i \\ &= \left(-\sum_{i<j} 4J_{ij}x_ix_j + \sum_{i<j} 2J_{ij}x_ix_i + \sum_{i<j} 2J_{ij}x_jx_j - \sum_{i=1}^{N} 2h_ix_ix_i\right) + \left(-\sum_{i<j} 4J_{ij} + \sum_{i=1}^{N} h_i\right) \\ & \qquad [\because \;\; x_i^2 = x_i \;\; \text{for binary variables}] \\ &= \sum_{i=1}^{N}\sum_{j=1}^{i} \mathcal{I}_{ij}x_ix_j + A \end{aligned}$$

where,

$$A = \sum_{i=1}^{N} h_i - \sum_{i<j} 4J_{ij} \tag{2.4}$$

$$\begin{aligned} \mathcal{I}_{ij} &= -4J_{ij} \;\; (\text{for} \;\; i \neq j) \\ &= -2h_i + \sum_{i<k} 2J_{ik} + \sum_{l>i} 2J_{il} \;\; (\text{for} \;\; i = j) \end{aligned} \tag{2.5}$$

The above calculation proves that the Ising model can be reformulated as a QUBO model and vice versa. The quadratic terms in the QUBO model are equivalent to two-body interactions in the Ising model Hamiltonian. We can represent many NP-hard problems as Ising formulations, as found in [22].

Chapter 3

Quantum Annealing and D-wave Systems

The term "annealing" stems from material science and is synonymous with "heat treatment", wherein a cycle of increasing and subsequently decreasing the temperature changes the physical and chemical properties of a material. It turns out that such a phenomenon can also be used to find heuristic solutions to optimization problems.

3.1 Simulated Annealing

An optimization problem can be interpreted in an energy framework in which each solution is a point on the energy landscape, and the desired solution is the global minima; now, annealing can be applied to this system to drive it to the minimum energy state. In the classical sense, a system can be prepared to mimic the required energy landscape followed by heating and subsequently cooling to make the system jump energy barriers and possibly reach a lower energy state in that iteration.

It is possible to simulate such a classical setting using classical computers; this method of solving problems is termed *Simulated Annealing*. In simulated annealing, a specific probability function $P(\mathbf{E_i}, \mathbf{E_{i+1}}, \mathbf{T})$ is formulated, which gives the probability for a system to move from the energy state $\mathbf{E_i}$ to the neighbouring state $\mathbf{E_{i+1}}$ at a temperature T. At lower values of T, the probability of the state moving downhill in the energy landscape increases, and as T approaches 0, the probabilities mimic a greedy algorithm which only moves downhill and eventually settles at a minima [18].

3.2 Adiabatic Quantum Computing and Quantum Annealing

Adiabatic quantum computing (AQC) is a quantum computation method involving perturbing a quantum system slowly to achieve the desired result at the end of the process. AQC at its core is based on the '*adiabatic theorem*' [16] in quantum mechanics which describes that a quantum system in an initial non-degenerate ground state of a time-dependent Hamiltonian will continue to remain in the ground state given that Hamiltonian changes sufficiently slowly. The run time of a specific task depends on the minimum eigenvalue (energy) gap between the ground and the first excited state, i.e., if the energy gap is high, the system must be perturbed slower for the adiabatic theorem to apply [17].

$$t_{run} = O\left(\frac{1}{(\Delta E_{min})^2}\right) \tag{3.1}$$

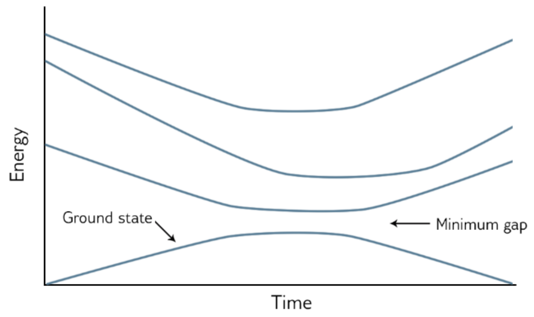


Figure 3.1: Energy diagram of various states as the system evolves with time Source: https://docs.dwavesys.com/docs/latest/c_gs_2.html#id2

Quantum Annealing (QA) is motivated by the same idea mentioned previously in simulated annealing but this time, it is done in a quantum system instead of a classical system. The major difference in quantum annealing is that the system does not have to cross the entire energy barrier of the energy landscape to travel between local/global minima but can move through the barrier using quantum tunnelling; hence it provides an advantage compared to the Hill-climb approach of classical annealing [19] [20] [21].

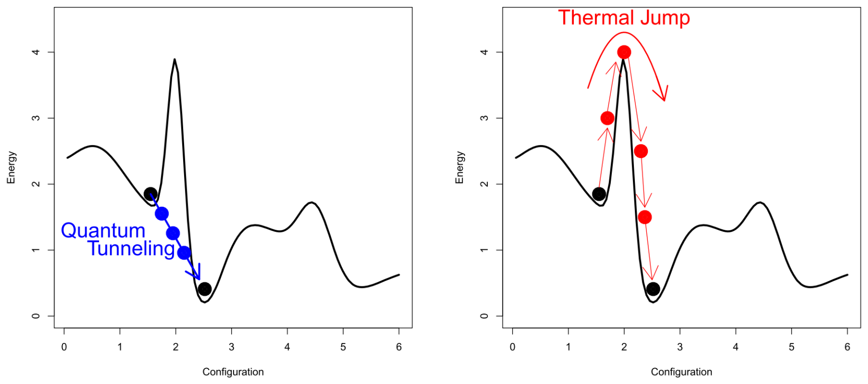


Figure 3.2: Quantum Annealing vs Classical Annealing. Source: [14]

The required Hamiltonian for quantum annealing of a problem can be described as

$$\mathcal{H} = A(s)\mathcal{H}_i + B(s)\mathcal{H}_f \tag{3.2}$$
$$\text{where, } s(t) : [0, t_{run}] \rightarrow [0, 1]$$

In the expression, $\mathcal{H}_i$ represents the initial Hamiltonian of the system, and $\mathcal{H}_f$ is the final Hamiltonian which encodes the problem to be optimized. The system is initially prepared to be in the ground state of $\mathcal{H}_i$ and at $t = 0\, A(s) >> B(s)$. As time evolves, the function $A(s)$ decreases and the function $B(s)$ increases; hence the contribution of the final Hamiltonian increases with time. As $s(t) \rightarrow 1\, B(s) >> A(s)$

Hence the system is driven towards the ground state of the final Hamiltonian and then measured to find the solution to the optimization problem. As any annealing process is probabilistic, this procedure is iterated multiple times and the state corresponding to the minimum energy is recorded. Solution states corresponding to higher energies are observed during some iterations due to external interference or thermal fluctuations in the system.

3.3 Quantum Annealing for Ising model/QUBO problems

Quantum annealing can be applied to solve for problems formulated in the Ising model. Once we come up with a method to solve Ising problems, it can also be applied to Qubo problems as we can convert between Ising and QUBO frameworks easily.

An Ising model consists of N variables, each representing a spin up or spin sown state represented by the values of 1 and -1, i.e., $S_i \in \{-1, 1\}$.

The energy of the Ising model can be given by its Hamiltonian 2.1. Our goal is to find such a configuration of spins such that this function is minimized; now our aim is to form appropriate initial and final Hamiltonians $\mathcal{H}_i$ and $\mathcal{H}_f$, which can be annealed.

Since we intend to work with qubits, we represent spins in terms of $|0\rangle$ or $|1\rangle$ and the Hamiltonians in terms of Pauli matrices. We need a $\mathcal{H}_i$ that is easy to prepare and couples the adjacent spins (leading to spin flips). This is achieved by such a Hamiltonian:

$$\mathcal{H}_i = -\sum_{i=1}^{N} \sigma_x^i \tag{3.3}$$

where, σ_x^i is the Pauli X matrix acting on the i^{th} qubit i.e.,

$$\left(\sigma_x^i = \mathbb{I} \otimes \ldots \otimes \sigma_x \otimes \ldots \mathbb{I}\right)$$

Since the Hamiltonian is a sum of Pauli X matrices, it is apparent that all eigenstates will be a combination of $|+\rangle$ and $|-\rangle$ states and the eigenstate corresponding to minimum eigenvalue is the $|+\rangle^N$ state with eigenvalue $-N$. Therefore, the initial ground state of the system is:

$$|\psi\rangle = \left(\frac{1}{\sqrt{2}}\right)^N (|0\rangle + |1\rangle) \otimes \ldots \otimes (|0\rangle + |1\rangle) \tag{3.4}$$

This state represents an equal superposition of all possible spin configurations of the Ising model.

The final Hamiltonian $\mathcal{H}_f$ of the system must encode within it the information in the Ising Hamiltonian $\mathcal{H}_{ising}$ and must have eigenstates corresponding to a combination of N qubits; each is either $|0\rangle$ or $|1\rangle$ state. Such a Hamiltonian can be achieved by replacing the S_i and S_j terms of 2.1 with corresponding Pauli Z matrices acting on the i^{th} and the j^{th} qubits.

$$\mathcal{H}_f = -\sum_{i,j=1}^{N} J_{ij}\sigma_z^i\sigma_z^j - \sum_{i=1}^{N} h_i\sigma_z^i \tag{3.5}$$

where, σ_z^i is the Pauli Z matrix acting on the i^{th} qubit i.e.,

$$\left(\sigma_z^i = \mathbb{I}\otimes\ldots\otimes\sigma_z\otimes\ldots\mathbb{I}\right)$$

By substitution of $\mathcal{H}_i$ (see 3.3) and $\mathcal{H}_f$ (see 3.5) into 3.2 we get a total Hamiltonian of the system as:

$$\mathcal{H} = A(s)\left(-\sum_{i=1}^{N}\sigma_x^i\right) + B(s)\left(-\sum_{i,j=1}^{N} J_{ij}\sigma_z^i\sigma_z^j - \sum_{i=1}^{N} h_i\sigma_z^i\right) \tag{3.6}$$

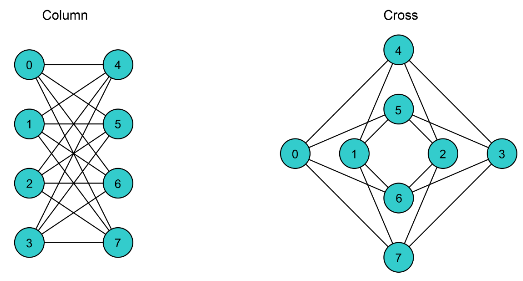
As this time-dependent Hamiltonian smoothly transitions between $\mathcal{H}_i$ and $\mathcal{H}_f$, there will be many spin-flips at the start of the annealing process and the number of spin flips decreases with time as the contribution of the initial Hamiltonian decreases. Each qubit will gradually collapse to either the $|0\rangle$ or the $|1\rangle$ state, depending on which configuration minimizes the Ising Hamiltonian $\mathcal{H}_{ising}$ initially formulated. Repeating this process multiple times, ideally under adiabatic conditions (slow enough) and recording the lowest energy eigenstates yields us the solution to the original optimization problem that we desired to solve.

3.4 Chimera Topology in D-Wave systems

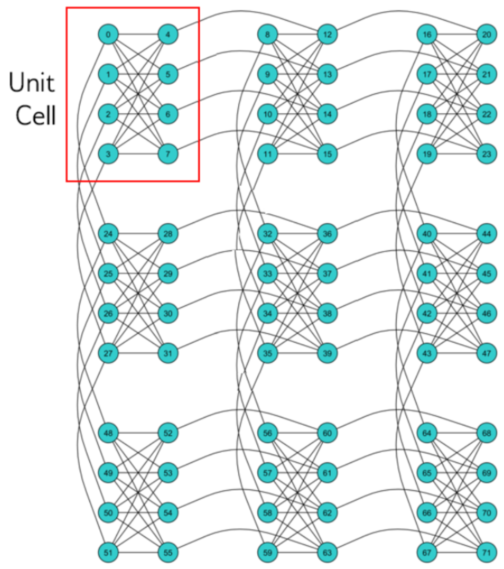
Now that all the mathematical groundwork is laid out, the next step is to actually create the required structure in a Quantum Processing Unit (QPU). For this purpose, the architecture of the QPU must be such that it is capable of encoding any optimization problem within it. D-Wave 2000Q systems use the Chimera Topology in their QPU design to achieve this goal.

The Chimera Graph is a lattice of many cells wherein each cell is a $K_{4,4}$ bipartite graph along with additional edges from each node to connect to nodes of the neighbouring cells.

The $K_{4,4}$ bipartite unit cell is achieved by laying out a 4x4 grid in each unit cell, each row and each column in that grid represents one qubit, and each intersection represents an edge of the bipartite graph, also called the *internal couplers* between qubits. The edges which connect neighbouring unit cells can be pictured as each row(qubit) being connected to the row of the same enumeration in the adjacent cell and similarly for the columns. These connections between two adjacent cells are called the *external couplers*.



(a) A



(b) B

Figure 3.3: A shows an alternate representation of K4,4 graph usually depicted in D-Wave systems. Fig. B represents the graphical structure of the Chimera Topology. Source - https://docs.ocean.dwavesys.com/en/stable/concepts/topology.html

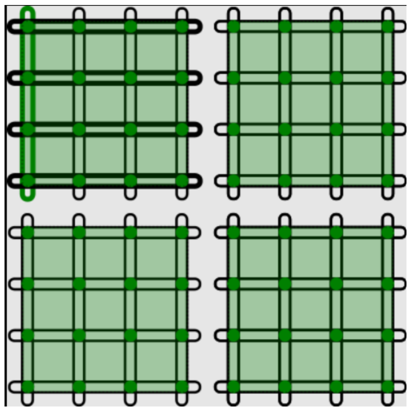


Figure 3.4: Qubit grid structure for Chimera graph, each green dot represents a coupler. Source -https://docs.ocean.dwavesys.com/en/stable/concepts/topology.html

3.5 Minor Embedding

To successfully encode a particular problem into the QPU, knowing which qubits to use and which couplers to activate between those qubits is essential. It is helpful to imagine each QUBO/Ising problem in terms of a graph where each node is a variable, and there is an edge between two nodes if they share a non-zero coupling constant. Thus, now all that is required is to find a subgraph within the Chimera graph which represents the graph of the problem.

However, now it can be seen that for the $K_{4,4}$ bipartite lattice used in the Chimera topology, it is impossible to find a subgraph as simple as a K_3 graph, i.e., a triangle. It can be seen that any complete graph $G \in K_i$ such that $i > 2$ cannot be a subgraph of the Chimera graph. Additionally, any problem graph with a subgraph that is K_i can also not be a subgraph. To overcome these constraints, the process of *Minor Embedding* is used.

Instead of finding subgraphs which are precisely equal to the problem graph, in Minor embedding, subgraphs which are homeomorphic to the problem graph are found, which can successfully be implemented on the Chimera graph. For implementing these homeomorphisms, multiple qubits are clubbed together to act as a single node of the problem graph. The qubits connected this way is called a *chain*, and a parameter called *chain strength* is set so that the solver returns the same value for the qubits representing a single node. Though small problems can be manually embedded, it becomes cumbersome for bigger problems. Fortunately, there is a library of tools called *minorminer* [15] that embeds the given problem to be submitted to the QPU.

Chapter 4

Simulation results

We have implemented three problems on the D-Wave quantum computers that can be expressed as a QUBO problem -

- The max-cut problem
- The minimum vertex cover problem (MVC)
- The travelling salesman problem (TSP)

4.1 Simple QUBO

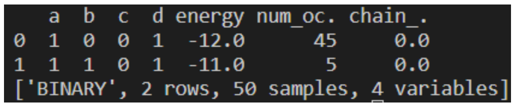
We first look at a very simple QUBO problem. Say we wanted to minimize the following objective function without any constraints

$$y = -6x_1 - 2x_2 - 7x_3 - 6x_4 + 3x_1x_2 + 2x_2x_3 + 7x_1x_3 + 11x_3x_4 \tag{4.1}$$

The code to solve the above problem is written as described below:

1. Import the necessary packages
2. We then set the linear and quadratic terms and define the Q matrix with these terms as inputs
3. We submit the problem to the solver with the number of runs, and the solution is printed on the screen

We obtain the following output:



```
   a  b  c  d energy num_oc. chain_.
0  1  0  0  1  -12.0      45     0.0
1  1  1  0  1  -11.0       5     0.0
['BINARY', 2 rows, 50 samples, 4 variables]
```

Figure 4.1: a, b, c, d are the variables; num_oc is the number of occurrences of a particular solution

We see that the solution to the above problem is $(x_1, x_2, x_3, x_4) = (1, 0, 0, 1)$ with the minimum value of -12

4.2 The Maximum cut problem

The Max-cut problem directly takes a QUBO formulation described before, see 1.1.1. We will implement the problem explained in the example, see 2. The chain strength is set to 8 to ensure that the qubits assigned to the same variable should return the same value; otherwise, the solution is infeasible. We also perform multiple runs to ensure that an optimal solution is found. The code is written as follows:

1. We first import the necessary packages
2. We initialize our graph and add edges according to our problem
3. We then set up our Q matrix and parameters such as chain strength, number of runs etc.
4. The QUBO is then submitted to the solver, and the result is stored in ‘response’.
5. The two sets are defined and printed on the screen
6. The ‘best’ result is then stored as a png file

The results obtained are as follows-

The outputs are shown in figure 4.3. We see that the solver was able to find minimum energy solutions, and the two sets are also shown. The number of occurrences of each solution is given in figure 4.4.

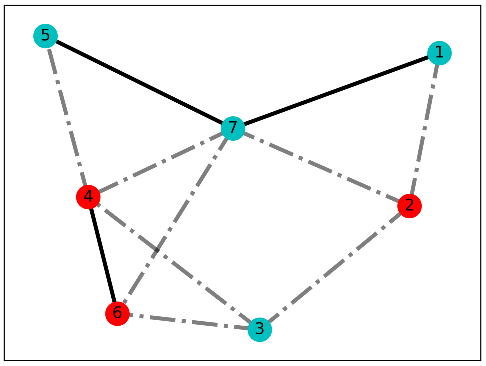
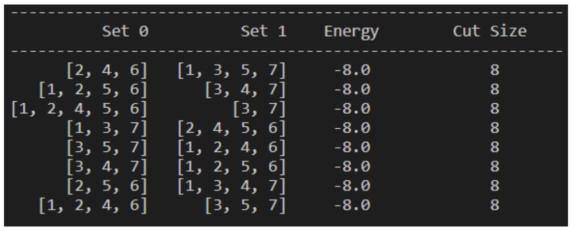
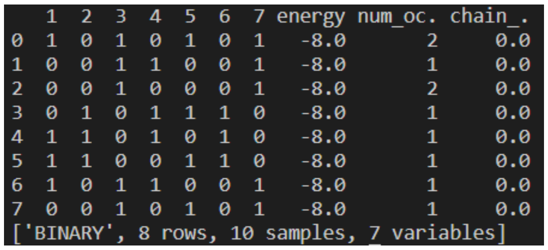


Figure 4.2: The nodes coloured in red are in one set and the ones coloured in cyan are in the other set; the solid lines are uncut and the dashed lines are cut



```
-----------------------------------------------------------------
           Set 0            Set 1    Energy          Cut Size
-----------------------------------------------------------------
       [2, 4, 6]    [1, 3, 5, 7]      -8.0                  8
    [1, 2, 5, 6]       [3, 4, 7]      -8.0                  8
 [1, 2, 4, 5, 6]          [3, 7]      -8.0                  8
       [1, 3, 7]    [2, 4, 5, 6]      -8.0                  8
       [3, 5, 7]    [1, 2, 4, 6]      -8.0                  8
       [3, 4, 7]    [1, 2, 5, 6]      -8.0                  8
       [2, 5, 6]    [1, 3, 4, 7]      -8.0                  8
    [1, 2, 4, 6]       [3, 5, 7]      -8.0                  8
```

Figure 4.3



```
   1  2  3  4  5  6  7 energy num_oc. chain_.
0  1  0  1  0  1  0  1   -8.0       2     0.0
1  0  0  1  1  0  0  1   -8.0       1     0.0
2  0  0  1  0  0  0  1   -8.0       2     0.0
3  0  1  0  1  1  1  0   -8.0       1     0.0
4  1  1  0  1  0  1  0   -8.0       1     0.0
5  1  1  0  0  1  1  0   -8.0       1     0.0
6  1  0  1  1  0  0  1   -8.0       1     0.0
7  0  0  1  0  1  0  1   -8.0       1     0.0
['BINARY', 8 rows, 10 samples, 7 variables]
```

Figure 4.4

4.3 Minimum vertex cover problem

We saw in 1.2.1 that the minimum vertex cover problem could be formulated as a QUBO problem using penalties to impose the constraints. We will seek to find the minimum vertex cover for the graph in example 2. Here again, we set the chain strength and the penalty to 8. We give ten runs to ensure an optimal solution. The code is similar to max-cut except in the definition of the Q matrix since we now have to include penalty terms too.

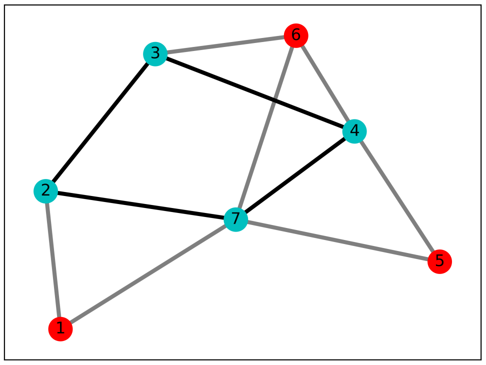
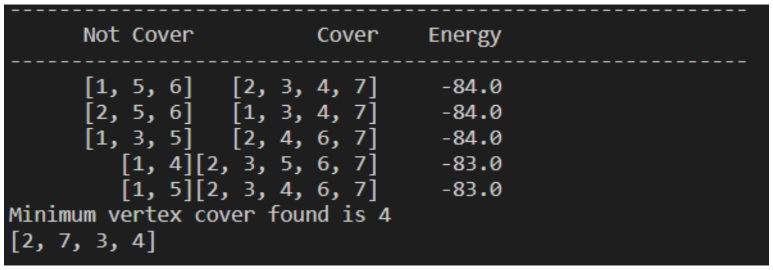
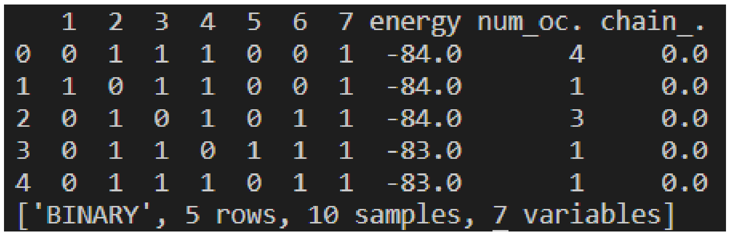


Figure 4.5: The cyan coloured vertices are included in the minimum cover while the red coloured vertices are not



```
----------------------------------------------------------------
      Not Cover            Cover     Energy
----------------------------------------------------------------
      [1, 5, 6]    [2, 3, 4, 7]      -84.0
      [2, 5, 6]    [1, 3, 4, 7]      -84.0
      [1, 3, 5]    [2, 4, 6, 7]      -84.0
         [1, 4][2, 3, 5, 6, 7]      -83.0
         [1, 5][2, 3, 4, 6, 7]      -83.0
Minimum vertex cover found is 4
[2, 7, 3, 4]
```

Figure 4.6



```
   1  2  3  4  5  6  7 energy num_oc. chain_.
0  0  1  1  1  0  0  1  -84.0       4     0.0
1  1  0  1  1  0  0  1  -84.0       1     0.0
2  0  1  0  1  0  1  1  -84.0       3     0.0
3  0  1  1  0  1  1  1  -83.0       1     0.0
4  0  1  1  1  0  1  1  -83.0       1     0.0
['BINARY', 5 rows, 10 samples, 7 variables]
```

Figure 4.7

4.4 Travelling salesman problem

The travelling salesman problem is one of the most well-known problems in optimization[6] [7]. There have been various attempts to solve the TSP using quantum algorithms- quantum approximate optimization algorithm(QAOA) [8] [9] and quantum annealing [10] [11]. We have formulated the TSP as a QUBO problem for four cities with the town layout given in figure 4.8.

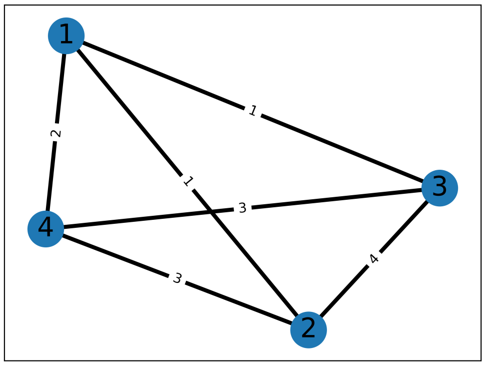


Figure 4.8: Town layout for the TSP. The nodes numbered 1-4 are the different towns. The numbers of the lines represent the distance between each pair of towns.

We solve for the node-based formulation as described in [12]. The objective Hamiltonian is given by:

$$H_O = \sum_{i=1}^{4} \sum_{\substack{u,v=1 \\ u \neq v}} C_{uv} x_u^i x_v^{i+1} \tag{4.2}$$

where,

$$\begin{aligned} C_{uv} &\rightarrow \text{the distance between cities u and v} \\ x_u^i &\rightarrow \text{the salesman is in the city } u \text{ at the } i^{th} \text{ step} \end{aligned}$$

The penalties that the salesman is in one and only one city at a given instant of time and that he visits each city is imposed by the following penalty Hamiltonian:

$$H_P = \sum_{v=1}^{4} \left(1 - \sum_{i=1}^{4} x_v^i\right)^2 + \sum_{i=1}^{4} \left(1 - \sum_{v=1}^{4} x_v^i\right)^2 \tag{4.3}$$

The first term imposes the constraint that the salesman visits each city, and the second term ensures that the salesman is in only one city at a given instant of time. We get the first term following the steps below (the second term is obtained similarly):

1. The constraint that the salesman visits each city is imposed by the equality

$$\sum_{i=1}^{4} x_v^i = 1$$

2. We define an objective function by subtracting one side of the equality from the other and taking the square [13]

$$\left(1 - \sum_{i=1}^{4} x_v^i\right)^2$$

3. We then sum over all the cities

$$\sum_{v=1}^{4}\left(1 - \sum_{i=1}^{4} x_v^i\right)^2$$

The total Hamiltonian is given by

$$H_T = H_O + PH_P \tag{4.4}$$

where P is the penalty constant set to 8 in our simulation.

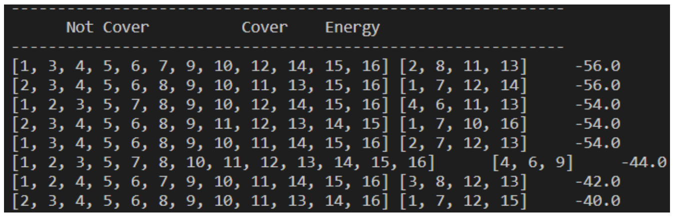
The code is written as described below:

1. Import the necessary packages
2. Create the weighted graph(town layout); the weights are the distances between cities
3. Set the QPU parameters- P, chain strength, number of runs etc.
4. Set up the Q matrix - it will be 16×16 since there are 16 variables.

   - The first loop is over time, and the second loop runs over all the vertices joined by edges in the graph.
   - The diagonal terms are set to $-2P$, and the terms representing the same vertex at different times are set to P
   - The terms representing acceptable transitions are set to the weights(distances between cities). For example, Q_{18} is set to 2, the weight of the edge between 1 and 4. (here Q_{18} means the salesman visits the city 1 followed by 4)
   - The terms corresponding to different vertices at the same time are set to P
   - The remaining terms are set to 0

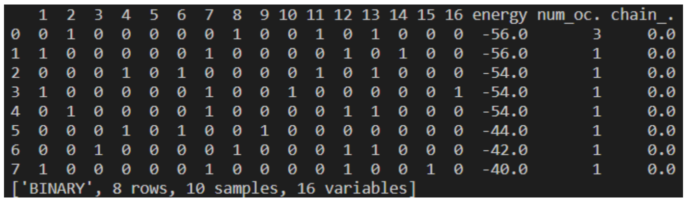
5. Run the code; the result is stored in 'S'
6. Print the results - the first set contains the variables that are 0, i.e. not part of the solution; the second set stores the solution to our problem
7. The town layout is stored in a png file

The results are given in figures 4.9 and 4.10



```
------------------------------------------------------------
      Not Cover             Cover     Energy
------------------------------------------------------------
[1, 3, 4, 5, 6, 7, 9, 10, 12, 14, 15, 16] [2, 8, 11, 13]     -56.0
[2, 3, 4, 5, 6, 8, 9, 10, 11, 13, 15, 16] [1, 7, 12, 14]     -56.0
[1, 2, 3, 5, 7, 8, 9, 10, 12, 14, 15, 16] [4, 6, 11, 13]     -54.0
[2, 3, 4, 5, 6, 8, 9, 11, 12, 13, 14, 15] [1, 7, 10, 16]     -54.0
[1, 3, 4, 5, 6, 8, 9, 10, 11, 14, 15, 16] [2, 7, 12, 13]     -54.0
[1, 2, 3, 5, 7, 8, 10, 11, 12, 13, 14, 15, 16]      [4, 6, 9]     -44.0
[1, 2, 4, 5, 6, 7, 9, 10, 11, 14, 15, 16] [3, 8, 12, 13]     -42.0
[2, 3, 4, 5, 6, 8, 9, 10, 11, 13, 14, 16] [1, 7, 12, 15]     -40.0
```

Figure 4.9



```
   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 energy num_oc. chain_.
0  0  1  0  0  0  0  0  1  0  0  1  0  1  0  0  0  -56.0       3     0.0
1  1  0  0  0  0  0  1  0  0  0  0  1  0  1  0  0  -56.0       1     0.0
2  0  0  0  1  0  1  0  0  0  0  1  0  1  0  0  0  -54.0       1     0.0
3  1  0  0  0  0  0  1  0  0  1  0  0  0  0  0  1  -54.0       1     0.0
4  0  1  0  0  0  0  1  0  0  0  0  1  1  0  0  0  -54.0       1     0.0
5  0  0  0  1  0  1  0  0  1  0  0  0  0  0  0  0  -44.0       1     0.0
6  0  0  1  0  0  0  0  1  0  0  0  1  1  0  0  0  -42.0       1     0.0
7  1  0  0  0  0  0  1  0  0  0  0  1  0  0  1  0  -40.0       1     0.0
['BINARY', 8 rows, 10 samples, 16 variables]
```

Figure 4.10

We see that the optimal solutions in this run are that the salesman visit the towns in the order:

- $2 \rightarrow 4 \rightarrow 3 \rightarrow 1$
- $1 \rightarrow 3 \rightarrow 4 \rightarrow 2$

One is just the reverse of the other.

Code availability

All codes are uploaded onto our Github Repo

Conclusions

The Quantum Computing/Annealing approach to solving QUBO problems is still an idea in its preliminary stages with a lot of research ahead of it, but undoubtedly it is also potentially groundbreaking. QUBO problems comprise a large section of optimization problems which are directly concerned with practical situations in the world. The Ising model gives us a very elegant way of trying to solve these optimization problems from a deterministic standpoint and solving them in a heuristic sense to yield practical solutions which are immensely applicable. As we see from Chapter 5 of the report the quantum approach has transcended the bounds of pure theory and now have a very physical form capable of solving real problems. Although quantum supremacy is hotly debated, in the near future Quantum computing can very well become the newer and better way to tackle optimization problems helping massive sections of society such as business, industry, research etc.

References

[1] Boros, E. and Hammer, P.L., 2002. Pseudo-boolean optimization. Discrete applied mathematics, 123(1-3), pp.155-225.

[2] Karp, Richard M. "Reducibility among combinatorial problems." Complexity of computer computations. Springer, Boston, MA, 1972. 85-103.

[3] Glover, Fred, Gary Kochenberger, and Yu Du. "A tutorial on formulating and using QUBO models." arXiv preprint arXiv:1811.11538 (2018)

[4] Kochenberger, Gary, et al. "The unconstrained binary quadratic programming problem: a survey." Journal of combinatorial optimization 28.1 (2014): 58-81.

[5] Anthony, Martin, et al. "Quadratic reformulations of nonlinear binary optimization problems." Mathematical Programming 162.1 (2017): 115-144.

[6] G. Gutin and A. Punnen, eds., "The Traveling Salesman Problem and its Variations". *Combinatorial Optimization, Kluwer Academic Press*, 2002.

[7] V. Chvatal, D. L. Applegate, R. E. Bixby, and W. J. Cook, "The Traveling Salesman Problem: A Computational Study". *Princeton Series in Applied Mathematics, Princeton University Press*, 2011.

[8] S. Hadfield, Z. Wang, E. G. Rieffel, B. O'Gorman, D. Venturelli, and R. Biswas, "Quantum approximate optimization with hard and soft constraints," in *Proceedings of the Second International Workshop on Post Moores Era Supercomputing*, pp. 15–21, 2017.

[9] S. Hadfield, Z. Wang, B. O'Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, "From the quantum approximate optimization algorithm to a quantum alternating operator ansatz," *Algorithms*, vol. 12, no. 2, p. 34, 2019.

[10] R. Marto˘n´ak, G. E. Santoro, and E. Tosatti, "Quantum annealing of the travelling-salesman problem," *Physical Review E*, vol. 70, no. 5, p. 057701, 2004.

[11] G. E. Santoro and E. Tosatti, "Optimization using quantum mechanics: quantum annealing through adiabatic evolution," *Journal of Physics A: Mathematical and General*, vol. 39,no. 36, p. R393, 2006.

[12] Salehi, Ö., Glos, A. and Miszczak, J.A. "Unconstrained binary models of the travelling salesman problem variants for quantum optimization" *Quantum Inf Process* **21**, 67 (2022).

[13] D-Wave docs URL: https://docs.dwavesys.com/docs/latest/c_gs_6.html

[14] Yazhen Wang. Shang Wu. Jian Zou. "Quantum Annealing with Markov Chain Monte Carlo Simulations and D-Wave Quantum Computers." *Statist. Sci.* 31 (3) 362 - 398, August 2016. https://doi.org/10.1214/16-STS560

[15] D-Wave docs URL: https://docs.ocean.dwavesys.com/projects/minorminer/en/latest/

[16] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, “Quantum computation by adiabatic evolution,” *arXiv preprint quant-ph/0001106*, 2000

[17] A. M. Childs, E. Farhi, and J. Preskill, “Robustness of adiabatic quantum computation,” *Physical Review A*, vol. 65, no. 1, p. 012322, 2001.

[18] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” *science*, vol. 220, no. 4598, pp. 671–680, 1983

[19] B. Apolloni, C. Carvalho, and D. De Falco, “Quantum stochastic optimization,” *Stochastic Processes and their Applications*, vol. 33, no. 2, pp. 233–244, 1989.

[20] T. Kadowaki and H. Nishimori, “Quantum annealing in the transverse Ising model,” *Physical Review E*, vol. 58, no. 5, p. 5355, 1998

[21] C. C. McGeoch,“Adiabatic quantum computation and quantum annealing: Theory and practice”, *Synthesis Lectures on Quantum Computing*, vol. 5, no. 2, pp. 1–93, 2014

[22] A. Lucas, “Ising formulations of many NP problems,” *Frontiers in Physics*, vol. 2, p. 5, 2014.