به نام خداوند جان و خرد

Quantum Optimization Algorithms

Mohammad Ali Jafarizadeh

University of Tabriz

October 17, 2024

Outline I

Outline II

§ 1: Introduction to Quantum Optimization

Introduction to Binary Optimization Problems

- Binary Optimization Problems:
  - Decision variables $x_i \in \{0, 1\}$ or $x_i \in \{-1, 1\}$.
  - The goal is to maximize or minimize an objective function.
  - Subject to constraints (optional) or unconstrained.
- General Form:

$$\min_{x \in \{0,1\}^n} f(x)$$

- Common applications: scheduling, portfolio optimization, set covering, etc.

Relevance of Binary Optimization in Quantum Computing

- Combinatorial Optimization: Many real-world problems in logistics, finance, and machine learning involve binary optimization.
- Quantum Binary Optimization: Quantum computers are uniquely suited for exploring the exponentially large solution spaces of binary optimization problems.
- Quantum Speedup: Quantum algorithms can leverage superposition and entanglement to explore multiple states in parallel, offering potential speedups over classical methods.

Classical vs. Quantum Binary Optimization

- Classical Methods:
    - Brute Force: Exhaustive search of the solution space (exponential time).
    - Branch-and-Bound: A divide-and-conquer approach, but can still be computationally expensive for large problems.
    - Heuristic Methods: Simulated annealing, genetic algorithms (do not guarantee optimal solutions).
- Quantum Approaches:
    - Quantum Annealing: Uses quantum tunneling to escape local minima.
    - QAOA (Quantum Approximate Optimization Algorithm): Hybrid quantum-classical algorithm designed for combinatorial optimization problems.
- Advantages: Quantum approaches offer the potential to explore large solution spaces more efficiently than classical algorithms.

Real-World Applications of Quantum Binary Optimization

- Logistics:
  - Vehicle routing problems.
  - Supply chain optimization.
- Finance:
  - Portfolio optimization.
  - Risk management.
- Machine Learning:
  - Feature selection.
  - Training of large models (e.g., neural networks).
- Artificial Intelligence: AI model tuning and optimization for decision-making tasks.

Summary of Quantum Binary Optimization

- Binary optimization problems are central to many real-world challenges in logistics, finance, and machine learning.
- Quantum binary optimization offers significant potential advantages due to the ability of quantum systems to explore large solution spaces in parallel.
- Quantum algorithms like quantum annealing and QAOA are promising methods for solving complex binary optimization problems, especially in the context of NP-hard problems.

§ 2: Binary and Quadratic Optimization Problems

Binary Optimization Problems

- Binary Optimization Problems involve decision variables $x_i \in \{0, 1\}$ or $x_i \in \{-1, 1\}$.
- The goal is to minimize or maximize an objective function subject to constraints.
- General Form:

$$\min_{x \in \{0,1\}^n} f(x)$$

- Examples: Max-Cut, Knapsack, Set Covering Problems.
- Challenges:
  - NP-hard: The number of possible solutions grows exponentially with the size of the problem.
  - Complex solution landscapes require advanced strategies for finding global optima.

Challenges in Binary Optimization

- Exponential Search Space: With n binary variables, the solution space has 2^n possibilities.
- Classical Approaches:
  - Heuristics (e.g., simulated annealing, genetic algorithms).
  - Exact methods like branch-and-bound or brute-force search are often impractical for large problems.
- Quantum Potential: Quantum algorithms can search large spaces more efficiently through superposition and entanglement.

Quadratic Unconstrained Binary Optimization (QUBO)

- QUBO is a specific class of binary optimization where the objective function is quadratic and unconstrained.
- Problem Formulation:

$$\min_{x \in \{0,1\}^n} x^T Q x + c^T x$$

  where:
  - Q is an $n \times n$ matrix representing interactions between variables.
  - c is a vector of linear coefficients.
- QUBO models a wide range of combinatorial optimization problems.

Importance of QUBO in Optimization

- Flexibility: QUBO can represent many types of problems, including:
  - Portfolio optimization in finance.
  - Feature selection in machine learning.
  - Resource allocation in telecommunications.
- Standard Form: The QUBO formulation is widely used because it can easily be mapped to quantum systems such as quantum annealers and gate-based models.
- Optimization Task: By solving QUBO efficiently, many real-world problems can be addressed.

Classical vs Quantum Approaches to Solving QUBO

- Classical Approaches:
  - Heuristic methods (e.g., simulated annealing, genetic algorithms) are often used to approximate solutions.
  - Exact methods (e.g., brute force, branch-and-bound) are computationally expensive.
- Quantum Approaches:
  - Quantum Annealing: Solves QUBO by evolving the system to the ground state of a quantum system.
  - QAOA (Quantum Approximate Optimization Algorithm): A hybrid quantum-classical algorithm designed for combinatorial problems.
  - Advantage: Quantum algorithms explore multiple states in parallel, offering speedups for large QUBO problems.

Advantages of Quantum Approaches for QUBO

- Parallelism: Quantum systems can explore a large number of possible solutions simultaneously due to superposition.
- Potential Speedups: In theory, quantum approaches like annealing or QAOA can outperform classical methods for certain NP-hard problems.
- Scalability: Quantum algorithms have the potential to scale more effectively for high-dimensional QUBO problems.
- Real-world applications of quantum optimization are still in the early stages but showing promise.

Summary of Binary and Quadratic Optimization Problems

- Binary Optimization Problems involve decision variables restricted to binary values and are often NP-hard.
- QUBO provides a flexible framework for a wide range of combinatorial optimization problems.
- Quantum algorithms offer the potential to solve large-scale QUBO problems faster than classical approaches through parallel exploration of solution spaces.

QUBO Definition

- The Unconstrained Quadratic Binary Optimization problem (QUBO) is:

$$\text{optimize } H(x) = x^T Q x$$

- Where:
    - x is an n-vector of binary variables,
    - Q is an $n \times n$ symmetric matrix of constants,
    - $H(x)$ is called the energy of a QUBO solution x.

Formula Expansion of QUBO

- The objective function in QUBO (i.e., the energy of a QUBO solution x) can be expanded as:

$$H(x) = \sum_{i=1}^{n} Q_{ii} x_i^2 + 2 \sum_{i=1}^{n} \sum_{j=i+1}^{n} Q_{ij} x_i x_j$$

- This shows the quadratic and pairwise interaction terms in the QUBO problem.

Some well-known applications of QUBO

- The QUBO model is highly flexible and can be applied to a wide variety of optimization problems.
- Some well-known applications include:
    - Maximum cut (Max-Cut)
    - SAT and Max Sat Problems
    - Spin Glass Problems
    - Graph Coloring Problems
    - Number Partitioning Problems
    - Maximum Independent Set Problems
    - Machine learning feature selection
    - Set Packing Problems
    - Graph partitioning
    - Quadratic Assignment Problems
    - Capital Budgeting Problems
    - Multiple Knapsack Problems
    - Task Allocation Problems (distributed computer systems)
    - Maximum Diversity Problems
    - P-Median Problems
    - Asymmetric and Symmetric Assignment Problems

Symmetric Matrix in QUBO

- The matrix Q in QUBO is symmetric, meaning that $Q_{ij} = Q_{ji}$ for all i and j.
- This property simplifies the representation and computation of the QUBO objective function.

Basic QUBO Problem Formulation

- Minimize/Maximize $H(x) = x^T Qx$: x binary
- For a symmetric matrix Q

$$x^T Qx = \sum_{i=1}^{n}\sum_{j=1}^{n} Q_{ij}x_i x_j$$

  where $x_i \in \{0,1\}$.
- In linear + quadratic form:

$$H(x) = x^T Qx = \sum_{i=1}^{n} Q_{ii}x_i + \sum_{i=1}^{n}\sum_{j=i+1}^{n} 2Q_{ij}x_i x_j$$

Matrix Representation in QUBO

- The QUBO formulation is widely used in combinatorial optimization.
- The binary nature of x simplifies many optimization problems.

Conclusion: Basic QUBO Problem Formulation

- The QUBO formulation involves minimizing or maximizing $H(x) = x^T Qx$ where x is a binary vector and Q is a symmetric matrix.
- Binary optimization problems have wide applications in combinatorial optimization and can be tackled using both classical and quantum approaches.

§ 2.1: Examples of QUBO Formulation

§ 2.1.1: The Max Cut Problem

The Max Cut Problem

- Given an undirected graph $G(V, E)$, the Max Cut problem seeks to partition V into two sets such that the number of edges between the two sets (the cut) is as large as possible.
- We can model this problem by introducing binary variables x_i, where:

$$x_i = \begin{cases} 1 & \text{if vertex } i \text{ is in one set,} \\ 0 & \text{if vertex } i \text{ is in the other set.} \end{cases}$$

- The quantity $(1 - x_i)(1 - x_j)$ identifies whether the edge (i, j) is in the cut.

Max Cut Problem Formulation

- The problem of maximizing the number of edges in the cut can be formulated as:

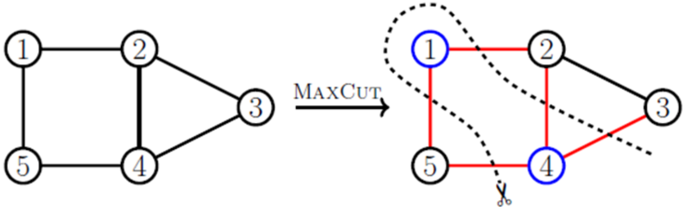
$$\text{maximize} \quad \sum_{(i,j)\in E} \frac{1 - x_i x_j}{2}$$

- This is an instance of a QUBO (Quadratic Unconstrained Binary Optimization) problem:

$$\text{maximize} \quad H(x) = x^T Q x$$

Example: The Max Cut Problem

- Consider an undirected graph with 5 vertices and 6 edges:
  - Vertices: 1, 2, 3, 4, 5
  - Edges: $(1,2),(1,3),(2,3),(3,4),(4,5),(2,5)$
- The objective is to partition the vertices into two sets such that the number of edges between the sets is maximized.



QUBO Representation of Max Cut Problem

- Explicitly taking into account all edges in the graph gives the following formulation:

$$Q = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

- This takes the desired form of a QUBO problem: maximize $H(x) = x^T Qx$.

Solving the Max Cut Problem

- Solving the QUBO model gives the optimal binary vector:

$$x = [0, 1, 1, 0, 0]$$

- Hence, vertices 2 and 3 are in one set, while vertices 1, 4, and 5 are in the other set.
- The maximum cut value for this problem is 5.

§ 2.1.2: The Number Partitioning Problem

The Number Partitioning Problem

- Partition a set of numbers into two subsets such that the subset sums are as close to each other as possible.
- We model this problem as a QUBO instance as follows:
- Consider a set of numbers $S = \{s_1, s_2, ..., s_n\}$.
- Let $x_i = 1$ if s_i is assigned to subset 1; 0 otherwise.
- The sum for subset 1 is given by $\sum_i x_i s_i$ and the sum for subset 2 is given by $\sum_i (1 - x_i) s_i$.

The Number Partitioning Problem

- We aim to minimize the difference between the sums of the two subsets:
$$\left(\sum_i x_i s_i - \sum_i (1-x_i) s_i\right)^2$$
- Simplifying the expression leads to:
$$\left(2\sum_i x_i s_i - \sum_i s_i\right)^2$$
- Dropping the additive and multiplicative constants, our QUBO optimization problem becomes:
$$\text{QUBO: } \min H(x) = \sum_{i,j} Q_{ij} x_i x_j$$

Numerical Example: The Number Partitioning Problem

- Consider the set of eight numbers:

$$S = \{25, 7, 13, 31, 42, 17, 21, 10\}$$

- From the previous development, the QUBO problem is:

$$\min H(x) = \sum_{i,j} Q_{ij} x_i x_j$$

with

$$Q = \begin{bmatrix} -3525 & 175 & 325 & 775 & 1050 & 425 & 525 & 250 \\ 175 & -1113 & 91 & 217 & 294 & 119 & 147 & 70 \\ 325 & 91 & -1989 & 403 & 546 & 221 & 273 & 130 \\ 775 & 217 & 403 & -4185 & 1302 & 527 & 651 & 310 \\ 1050 & 294 & 546 & 1302 & -5208 & 714 & 882 & 420 \\ 425 & 119 & 221 & 527 & 714 & -2533 & 357 & 170 \\ 525 & 147 & 273 & 651 & 882 & 357 & -3045 & 210 \\ 250 & 70 & 130 & 310 & 420 & 170 & 210 & -1560 \end{bmatrix}$$

Numerical Example: The Number Partitioning Problem

- Solving the QUBO gives $x = (00011001)$, yielding perfectly matched subset sums of 83.
- The development employed here can be expanded to address other forms of the number partitioning problems as discussed in Alidaee, et.al. (2005)

§ 3: Creating QUBO Models Using Known Penalties

Valid Infeasible Penalty (VIP)

- A penalty function is a **Valid Infeasible Penalty** (VIP) if:
  - It is zero for feasible solutions.
  - It is positive for infeasible solutions.
- Including quadratic VIPs in the objective function for each constraint yields a transformed QUBO model.

Creating QUBO using Known Penalties

- Many constrained problems can be re-formulated as QUBO models by introducing quadratic penalties with a positive scalar P.

Table: Simple examples: Known constraint/penalty pairs

Classical Constraint	Equivalent Penalty
$x + y \leq 1$	$P(xy)$
$x + y \geq 1$	$P(1 - x - y + xy)$
$x + y = 1$	$P(1 - x - y + 2xy)$
$x \leq y$	$P(x - xy)$
$x_1 + x_2 + x_3 \leq 1$	$P(x_1x_2 + x_1x_3 + x_2x_3)$
$x = y$	$P(x + y - 2xy)$

QUBO Models for Constrained Problems

- Certain types of constraints can be represented by quadratic penalty functions.
- For example, consider binary variables x and y with a constraint $x + y \leq 1$.
- A quadratic infeasibility penalty for this constraint is:

$$Pxy$$

where P is a large positive scalar.

§ 3.0.1: Minimum Vertex Cover (MVC) Problem

Minimum Vertex Cover (MVC) Problem

- A vertex cover is a subset of vertices such that every edge is incident to at least one vertex in the subset.
- The Minimum Vertex Cover (MVC) problem seeks to find the smallest such subset.
- The objective is to minimize the number of vertices in the cover.

$$\min \quad \sum_{j \in V} x_j$$

$$x_i + x_j \geq 1 \quad \text{for all} \quad (i, j) \in E$$

MVC Penalty Function

- The constraints in MVC can be represented by a penalty function:

$$P(1 - x - y + xy)$$

- This transforms the constrained problem into an unconstrained QUBO model as follows

$$\min \quad H(x) = \sum_{j \in V} x_j + P\left(\sum_{(i,j) \in E} 1 - x_i - x_j + x_i x_j\right)$$

- we can write this as minimize $\mathbf{x}^T Q \mathbf{x}$ plus a constant term

MVC Numerical Example

- Consider a graph with 6 edges and 5 nodes.
- The QUBO model is written as:

$$
\begin{aligned}
\min Qx = & x_1 + x_2 + x_3 + x_4 + x_5 + \\
& P(1 - x_1 - x_2 + x_1x_2) + \\
& P(1 - x_1 - x_3 + x_1x_3) + \\
& P(1 - x_2 - x_4 + x_2x_4) + \\
& P(1 - x_3 - x_4 + x_3x_4) + \\
& P(1 - x_3 - x_5 + x_3x_5) + \\
& P(1 - x_4 - x_5 + x_45)
\end{aligned}
$$

where Q is a matrix. Qx can be written as
$Qx = (1 - 2P)x_1 + (1 - 2P)x_2 + (1 - 3P)x_3 + (1 - 3P)x_4 + (1 - 2P)x_5 + Px_1x_2 + Px_1x_3 + Px_2x_4 + Px_3x_4 + Px_3x_5 + Px_4x_5 + 6P$

MVC Solution Example

- Arbitrarily choosing $P = 8$ and dropping the additive constant gives a QUBO model, with the Q matrix given by

$$\begin{bmatrix} -15 & 4 & 4 & 0 & 0 \\ 4 & -15 & 0 & 4 & 0 \\ 4 & 0 & -23 & 4 & 4 \\ 0 & 4 & 4 & -23 & 4 \\ 0 & 0 & 4 & 4 & -15 \end{bmatrix}$$

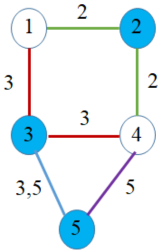
- The solution to the QUBO model is:

$$x = (01101), \quad x^T Qx = -45$$

- The minimum vertex cover consists of nodes 2, 3, and 5.

Numerical Example

- Solving this QUBO model gives: $H(x) = x^T Qx = -45$ at $x = (0, 1, 1, 0, 1)$ for which $H(x) = 48 - 45 = 3$, meaning that a minimum cover is given by nodes 2, 3, and 5.



§ 3.0.2: The Set Packing Problem

The Set Packing Problem

- The Set Packing Problem maximizes the number of selected disjoint subsets.
- This can be formulated as:

$$\max \sum w_i x_i \quad \text{subject to} \quad \sum a_{ij} x_i \leq 1$$

where x_i are binary variables, and w_i and a_{ij} are weights and coefficients.

Set Packing as QUBO

- By applying penalties, we can reformulate the Set Packing problem as a QUBO:
$$\max H(x) = x^T Q x$$
where the matrix Q is constructed using penalties for violating the disjoint constraint.

Set Packing Numerical Example

- Consider a small example of a set packing problem:

$$\max \; x_1 + x_2 + x_3 + x_4$$
$$s.t. \; x_1 + x_3 + x_4 \leq 1, \; x_1 + x_2 \leq 1$$

Re-casting as QUBO via the penalties of previous Table.

$$\max \; x_1 + x_2 + x_3 + x_4 - Px_1x_3 - Px_1x_4 - Px_3x_4 - Px_1x_2$$

- This has our customary QUBO form $\max H(x) = x^T Qx$, where the Q matrix , with P arbitrarily chosen to be 6, is given by

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

Set Packing Numerical Example

- The solution of this QUBO model is:

$$x = (0, 1, 1, 0), \quad y = 2$$

- All penalty terms are equal to zero in the solution.
- Remark: Set packing problems with thousands of variables and constraints have been efficiently reformulated and solved in Alidaee, et. al. (2008) using the QUBO reformulation illustrated in this example.

§ 3.0.3: The Graph Coloring Problem

The Graph Coloring Problem

- Vertex coloring problems seek to assign colors to nodes of a graph such that adjacent nodes receive different colors.
- These problems can be modeled as satisfiability problems as follows:
- Let $x_{ij} = 1$ if node i is assigned color j and 0 otherwise.
- Each node must be assigned a color, so we have the constraints:

$$\sum_{j=1}^{K} x_{ij} = 1 \quad \text{for all nodes } i$$

- For adjacent nodes i and j, the constraints ensure that they receive different colors.

Transformation into QUBO Model

- A feasible coloring where adjacent nodes are assigned different colors is assured by imposing adjacency constraints.
- This problem can be recast into a QUBO model using:
  - Transformation #1 on the node assignment constraints.
  - Transformation #2 on the adjacency constraints.
- The resulting QUBO model is optimized to find the feasible coloring.

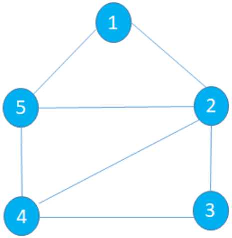
Numerical Example: Graph Coloring with 3 Colors

- Consider the problem of finding a feasible coloring of the graph using $K = 3$ colors.
- The goal is to find a solution to the system:

$$\sum_{j=1}^{3} x_{ij} = 1 \quad \text{for all adjacent nodes } i \text{ and } j$$

such that $x_{ip} + x_{jp} \leq 1$ for $p = 1, \ldots, 3$ for all adjacent nodes i and j.

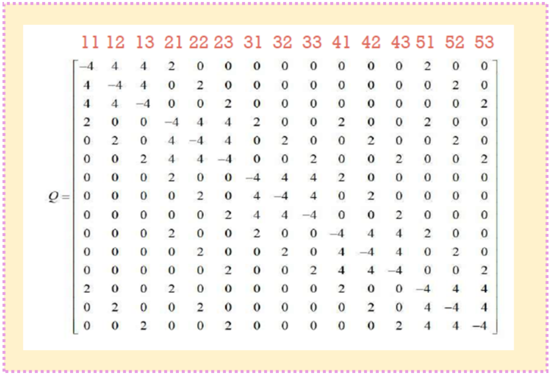
- Graph structure:



Solving the QUBO Model for Graph Coloring

- The problem is formulated as a QUBO problem:

$$\text{minimize } H(x) = x^T Q x \quad \text{where } x \text{ is binary}$$



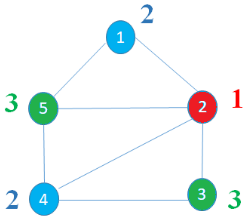
$$Q = \begin{array}{ccccccccccccccc}
11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 & 41 & 42 & 43 & 51 & 52 & 53 \\
-4 & 4 & 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\
4 & -4 & 4 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\
4 & 4 & -4 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\
2 & 0 & 0 & -4 & 4 & 4 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 \\
0 & 2 & 0 & 4 & -4 & 4 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 \\
0 & 0 & 2 & 4 & 4 & -4 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 \\
0 & 0 & 0 & 2 & 0 & 0 & -4 & 4 & 4 & 2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 4 & -4 & 4 & 0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 4 & 4 & -4 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & -4 & 4 & 4 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 4 & -4 & 4 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 4 & 4 & -4 & 0 & 0 & 2 \\
2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & -4 & 4 & 4 \\
0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 4 & -4 & 4 \\
0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 4 & 4 & -4
\end{array}$$

Solving the QUBO Model for Graph Coloring

- Solving this model yields a feasible coloring:

$$x_2 = x_4 = x_9 = x_{11} = x_{15} = 1$$

  with all other variables equal to zero.
- In this example, the solution means:
  - Nodes 1 and 4 get color #2.
  - Node 2 gets color #1.
  - Nodes 3 and 5 get color #3.



§ 4: The Max 2-Sat Problem

Introduction to Max 2-Sat Problem

- Satisfiability problems are used in many settings and represented in terms of clauses in conjunctive normal form (CNF).
- Max 2-Sat problems consist of clauses with two literals, and a clause is satisfied if either or both literals are true.
- The goal is to maximize the number of satisfied clauses.

Quadratic Penalties for Clause Types

- There are three types of clauses in Max 2-Sat, each associated with a quadratic penalty:
  1. **No negations**: $x_i \vee x_j$
     $$\text{Constraint: } x_i + x_j \geq 1, \quad \text{Penalty: } 1 - x_i - x_j + x_i x_j$$
  2. **One negation**: $x_i \vee \overline{x}_j$
     $$\text{Constraint: } x_i + (1 - x_j) \geq 1, \quad \text{Penalty: } x_j - x_i x_j$$
  3. **Two negations**: $\overline{x}_i \vee \overline{x}_j$
     $$\text{Constraint: } (1 - x_i) + (1 - x_j) \geq 1, \quad \text{Penalty: } x_i x_j$$

QUBO Model for Max 2-Sat

- The problem is recast into a Quadratic Unconstrained Binary Optimization (QUBO) model by minimizing the number of unsatisfied clauses.
- The penalty function is:

$$\min H(x) = x^T Qx$$

- Example: A Max 2-Sat instance with 4 variables and 12 clauses gives the QUBO model:

$$\min H(x) = 3 + x_1 - 2x_4 - x_2x_3 + x_2x_4 + 2x_3x_4$$

QUBO Model for Max 2-Sat

Table: Penalties for the 12 Clauses

Clause #	Clause	Quadratic Penalty
1	$x_1 \vee x_2$	$1 - x_1 - x_2 + x_1 x_2$
2	$x_1 \vee \overline{x}_2$	$x_2 - x_1 x_2$
3	$\overline{x}_1 \vee x_2$	$x_1 - x_1 x_2$
4	$\overline{x}_1 \vee \overline{x}_2$	$x_1 x_2$
5	$\overline{x}_1 \vee x_3$	$x_1 - x_1 x_3$
6	$\overline{x}_1 \vee \overline{x}_3$	$x_1 x_3$
7	$x_2 \vee \overline{x}_3$	$x_3 - x_2 x_3$
8	$x_2 \vee x_4$	$1 - x_2 - x_4 + x_2 x_4$
9	$\overline{x}_2 \vee x_3$	$x_2 - x_2 x_3$
10	$\overline{x}_2 \vee \overline{x}_3$	$x_2 x_3$
11	$x_3 \vee x_4$	$1 - x_3 - x_4 + x_3 x_4$
12	$\overline{x}_3 \vee \overline{x}_4$	$x_3 x_4$

QUBO Matrix

The QUBO model can be represented in matrix form as:

$$\min H(x) = 3 + x^T Q x$$

where the Q matrix is:

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} \\ 0 & -\frac{1}{2} & 0 & 1 \\ 0 & \frac{1}{2} & 1 & -2 \end{pmatrix}$$

Conclusion

- Solving the QUBO model minimizes the number of unsatisfied clauses.
- Example solution: $H(x) = 1$ when $x_1 = x_2 = x_3 = 0$ and $x_4 = 1$, meaning all but one clause is satisfied.
- The QUBO approach is scalable and has been used to solve Max 2-Sat problems with hundreds of variables.

Remarks on the QUBO Approach

- The QUBO model size is independent of the number of clauses and depends only on the number of variables.
- Max 2-Sat problems with hundreds of variables and thousands of clauses can be efficiently solved using the QUBO approach.

§ 5: Quantum Algorithms for Optimization

Contents I

Contents II

§ 5.1: Introduction to Quantum Optimization Algorithms

Introduction to Quantum Optimization Algorithms

- **Quantum Optimization Algorithms** aim to solve complex optimization problems by utilizing the principles of quantum mechanics.
- **Quantum Adiabatic Optimization (AQO)**:
  - A form of optimization where the system evolves slowly from an initial simple Hamiltonian H_B to a final Hamiltonian H_P that encodes the problem solution.
  - The system stays in its ground state, finding the global minimum, provided the evolution is slow enough.
- Other quantum optimization techniques:
  - **Quantum Annealing (QA)**: A practical implementation of AQO used in hardware like D-Wave to solve optimization problems.
  - **Variational Quantum Algorithms (VQA)**: Combines quantum state preparation with classical optimization methods (e.g., VQE, QAOA).
- **Applications**: Used for solving problems like **Max-Cut**, and other combinatorial optimization problems.

§ 6: Mapping Optimization Problems to Quantum Hardware

QUBO to Ising Model Mapping

- QUBO (Quadratic Unconstrained Binary Optimization) is a mathematical formulation of optimization problems that can be efficiently mapped to quantum hardware.
- Ising Model: A physical model used in quantum systems to represent binary variables as spins $s_i \in \{-1, 1\}$.
- Transformation: The mapping of QUBO to the Ising model is done via the relation:

$$x_i = \frac{1 + s_i}{2}$$

  where $x_i \in \{0, 1\}$ represents binary variables, and $s_i \in \{-1, 1\}$ represents spin variables.

Significance of the Ising Model in Quantum Hardware

- The Ising model serves as a bridge between optimization problems and quantum systems.
- Hamiltonian Formulation: The Ising model is expressed as a Hamiltonian:

$$H = \sum_{i,j} J_{ij} s_i s_j + \sum_i h_i s_i$$

  where J_{ij} represents the interaction between spins and h_i represents an external magnetic field. Both J_{ij} and h_i can be derived in terms of Q_{ii} and Q_{ij}.
- Substituting s_i with $\sigma_z^{(i)} = I_{2i-1} \otimes \sigma_z \otimes I_{2n-i}$, the Ising problem is converted to quantum Ising Hamiltonian.

Ising Hamiltonian

- The Hamiltonian is often expressed as an Ising Hamiltonian:

$$H = \sum_{i<j} J_{ij}\sigma_i^z \sigma_j^z + \sum_i h_i \sigma_i^z$$

- σ^z is the Pauli Z operator

Significance of the Ising Model in Quantum Hardware

- Many quantum optimization algorithms, such as Quantum Adiabatic Optimization, and Quantum Annealing, operate by finding the ground state of this Ising Hamiltonian.
- The Ising model is used in quantum hardware like D-Wave to solve optimization problems.

§ 7: Quantum Adiabatic Optimization

Introduction to Adiabatic Quantum Computation

- Quantum computations can be implemented by the adiabatic evolution of a system's Hamiltonian.
- The system is initialized in the ground state of a simple Hamiltonian and adiabatically evolved to a Hamiltonian that encodes the solution.
- Adiabatic quantum computation (AQC) evolves Hamiltonians rather than applying quantum gates.

Adiabatic Theorem & Optimization Algorithm

- Adiabatic theorem: A quantum system remains in its ground state if the Hamiltonian evolves slowly.
- Evolution starts with H_{initial}, transitioning to H_{final} which encodes the solution.
- System evolution:

$$H(t) = (1 - s(t))H_{\text{initial}} + s(t)H_{\text{final}}$$

- Steps: Initialize in ground state of H_{initial}, slowly evolve to H_{final}, measure to obtain the solution.

Applications & Real-World Examples & Challenges

- Effective in solving NP-hard problems such as Max-Cut, Traveling Salesman, and Graph Coloring.
- Real-world applications: logistics (vehicle routing, supply chain), finance (portfolio optimization), and machine learning (feature selection).
- Implementations: D-Wave quantum systems perform adiabatic optimization.
- Challenges: Decoherence, noise, scalability, and time constraints (slow evolution required).

The Adiabatic Theorem

- **Schrodinger equation**:

$$i\hbar \frac{d}{dt}|\psi(t)\rangle = H(t)|\psi(t)\rangle$$

- **Instantaneous eigenstate**:

$$H(t)|n(t)\rangle = E_n(t)|n(t)\rangle$$

- **Initial condition**:

$$|\psi(0)\rangle = |n(0)\rangle$$

- If evolution is slow enough, the system remains in its instantaneous eigenstate.

$$|\psi(t)\rangle \approx e^{i\theta(t)}|\psi(0)\rangle, \qquad |\psi_n(t)\rangle = U(t)|\psi(0)\rangle,$$

$$U_I(t) = \sum_{q=0}^{\infty}(-1)^q \int_0^t dt_q \cdots \int_0^{t_2} dt_1 H_I(t_q)\cdots H_I(t_1)$$

Adiabatic Theorem (Born and Folk 1928)

- A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum.
- Under a slowly changing Hamiltonian $H(t)$, with instantaneous eigenstate $|n(t)\rangle$ and the corresponding energy $E_n(t)$, a quantum system evolves from an initial state $|n(0)\rangle$ to the final state $|n(t)\rangle$.

$$|\psi(0)\rangle = \sum_n c_n(0)|n(0)\rangle \xrightarrow[\text{State}]{\text{Final}} |\psi(t)\rangle = \sum_n c_n(t)|n(t)\rangle$$

$$c_n(t) = c_n(0)e^{i\theta_n(t)}e^{i\gamma_n(t)}$$

- Dynamical Phase: $\theta_n(t) = -\frac{1}{\hbar}\int_0^t E_n(t^{'})dt^{'}$
- Geometrical Phase: $\gamma(t) = i\int_0^t \langle n(t^{'})|\dot{n}(t^{'})\rangle dt^{'}$

Adiabatic Theorem Continued

- The rate of change of Hamiltonian $\frac{dH(t)}{dt}$ is small, and there is a finite gap $E_m(t) - E_n(t) \neq 0$ between energies $m \neq n \rightarrow$

$$\langle n(t^{'})|\dot{n}(t^{'})\rangle = -\frac{\langle m(t)|\dot{H}(t)|n(t)\rangle}{E_m(t) - E_n(t)} \rightarrow 0$$

- $|c_n(t)|^2 = |c_n(0)|^2$ so if the system begins in an eigenstate of $H(0)$, it remains in an eigenstate of $H(t)$ during the evolution, with a change of phase only.

Adiabatic Approximation

$$H(t)|n(t)\rangle = E_n(t)|n(t)\rangle$$

- Assume $m \neq n$ and perform the inner product with $\langle m(t)|$:

$$H(t)|m(t)\rangle = E_m(t)|m(t)\rangle, \qquad \langle m(t)|n(t)\rangle$$

$$\langle m(t)|\dot{n}(t)\rangle = -\frac{\langle m(t)|\dot{H}(t)|n(t)\rangle}{E_m(t) - E_n(t)}$$

- **Adiabatic approximation**: The rate of change in the Hamiltonian $\frac{dH(t)}{dt}$ is small, and there is a finite gap $E_m(t) - E_n(t)$.

Adiabatic Limit

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = H(t)|\psi(t)\rangle \rightarrow |\psi(t)\rangle = \sum_n c_n(t)|n(t)\rangle$$

$$i\hbar\dot{c}_m(t) + i\hbar\sum_n c_n(t)\langle m(t)|\dot{n}(t)\rangle = c_m(t)E_m(t)$$

- In the Adiabatic limit $\langle m(t)|n(t)\rangle \approx 0$ for $m \neq n$:

$$i\hbar\dot{c}_m(t) + i\hbar c_m(t)\langle m(t)|\dot{m}(t)\rangle = c_m(t)E_m(t)$$

$$\frac{\dot{c}_m(t)}{c_m(t)} = -\frac{i}{\hbar}E_m(t) + ii\langle m(t)|\dot{m}(t)\rangle$$

Adiabatic Limit

- In the Adiabatic limit $\langle m(t)|n(t)\rangle \approx 0$ for $m \neq n$:

$$c_m(t) = c_m(0)e^{i\theta_m(t)}e^{i\gamma_m(t)}$$

- **Dynamical phase**:

$$\theta_m(t) = -\frac{1}{\hbar}\int_0^t E_m(t^{'})dt^{'},$$

- **Geometrical phase**:

$$\gamma(t) = i\int_0^t \langle m(t^{'})|\dot{m}(t^{'})\rangle dt^{'}$$

§ 7.1: Quantum Adiabatic Algorithm

Quantum Adiabatic Algorithm

- The Quantum Adiabatic Algorithm (QAA) can be used on a quantum computer as an optimization method for finding the global minimum of a classical cost function $f : \{0,1\}^n \to \mathbb{R}$.
- The cost function is encoded in a problem Hamiltonian H_P, which acts on the Hilbert space of n spin-$\frac{1}{2}$ particles.

$$H_P = \sum_{z \in \{0,1\}^n} f(z)|z\rangle\langle z|$$

where for QUBO we have $H_p = \sum_{i<j} J_{ij}\sigma_i^z\sigma_j^z + \sum_i h_i\sigma_i^z$.

Initialization

- The system is initialized in the ground state of the beginning Hamiltonian H_B:

$$H_B = \sum_{i=1}^{n} \left(\frac{1 - \sigma_i^x}{2} \right)$$

- The ground state of H_B is the uniform superposition of computational basis states:

$$|\psi_{\text{init}}\rangle = \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} |z\rangle$$

Adiabatic Evolution

- The system evolves according to:

$$H(t) = (1 - t/T)H_B + (t/T)H_P$$

- Alternatively, using a parameter $s = t/T$, we write:

$$H(s) = (1 - s)H_B + sH_P$$

- Start in the ground state of H_B at $t = 0$ and evolve to $t = T$, where the state will be very close to the ground state of H_P, which gives the solution.

The Size of the Minimum Gap

- The minimum gap $g_{\min}$ is defined as:

$$g_{\min} = \min_{0 \leq s \leq 1} (E_1(s) - E_0(s))$$

- If $g_{\min} > 0$, the system evolves adiabatically, and the ground state is preserved.
- To ensure the system evolves correctly, the total evolution time T must satisfy:

$$T \gg \frac{E}{g_{\min}^2}$$

  where:

$$E = \max_{0 \leq s \leq 1} \left| \langle 1; s | \frac{dH}{ds} | 0; s \rangle \right|$$

Success Probability

- For a problem instance with a unique minimizing string w, the success probability at time $t = T$ is:

$$P(T) = |\langle w|\psi(T)\rangle|^2$$

Conclusion

- The strategies presented, including evolving more rapidly, initializing in excited states, and using modified evolution paths, consistently increased success probabilities in all hard instances tested.
- Future work will involve testing these strategies on larger problem instances using quantum hardware.

§ 8: One Qubit Example

One Qubit Problem Setup

- Consider a one-bit problem where the clause is satisfied if $z_1 = 1$.
- The problem Hamiltonian is given by:

$$H_P = \frac{1}{2} + \frac{1}{2}\sigma_z^{(1)}$$

- The ground state of H_P is $|z_1 = 1\rangle$.

The Initial Hamiltonian

- The initial Hamiltonian H_B is chosen as:

$$H_B = \frac{1}{2} - \frac{1}{2}\sigma_x^{(1)}$$

- The smooth interpolating Hamiltonian $H(s)$ is:

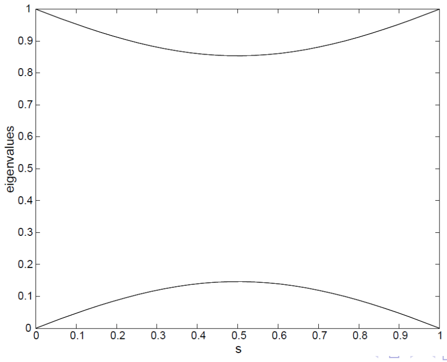
$$H(s) = (1 - s)H_B + sH_P$$

Eigenvalues of the Interpolating Hamiltonian

- The eigenvalues of $H(s)$ are given by:

$$\lambda_{\pm}(s) = \frac{1}{2}\left(1 \pm \sqrt{1 - 2s + 2s^2}\right)$$

- These eigenvalues are plotted in the following figure, showing a modest minimum gap:



Level Crossing with Symmetry

- Suppose we replace H_B with:

$$H'_B = \frac{1}{2} - \frac{1}{2}\sigma_z^{(1)}$$

- Now $H(s)$ is diagonal in the z-basis for all s, and the eigenvalues cross at $s = 0.5$:

$$\lambda_1 = s, \quad \lambda_2 = 1 - s$$

- This symmetry causes a level crossing and a minimum gap of zero.

Breaking the Symmetry

- Adding a small off-diagonal term to break the symmetry:

$$H_\epsilon(s) = \begin{pmatrix} s & \epsilon(1-s) \\ \epsilon(1-s) & 1-s \end{pmatrix}$$

- The minimum gap g_{min} is now ϵ, ensuring no level crossing.

§ 9: 2-SAT on a Ring

2-SAT on a Ring: Problem Setup

- Consider an n-bit problem with n clauses, each acting on adjacent bits.
- Clause C_j acts on bits j and $j+1$.
- The boundary condition is cyclic: bit $n+1$ is identified with bit 1.
- Clauses can either be "agree" (00 and 11 are satisfying) or "disagree" (01 and 10 are satisfying).
- The problem is restricted to an even number of disagree clauses so that a satisfying assignment exists.

Satisfying Assignment

- Given the list of clauses, constructing the satisfying assignment is trivial.
- If $w_1, w_2, \ldots, w_n$ is a satisfying assignment, so is $\overline{w}_1, \overline{w}_2, \ldots, \overline{w}_n$.
- There are exactly two satisfying assignments.

Problem Hamiltonian H_P

- The quantum version of the problem is described by the Hamiltonian:

$$H_P = H_{12}^{C_1} + H_{23}^{C_2} + \cdots + H_{n1}^{C_n}$$

- Each C_j is either an "agree" or "disagree" clause.
- The ground states of H_P are $|w_1\rangle|w_2\rangle\cdots|w_n\rangle$ and $|\overline{w}_1\rangle|\overline{w}_2\rangle\cdots|\overline{w}_n\rangle$.

Transformation of Hamiltonian

- Define the unitary transformation:

$$|z_1\rangle|z_2\rangle \dots |z_n\rangle \to |z'_1\rangle|z'_2\rangle \dots |z'_n\rangle$$

$$z'_j = \begin{cases} z_j & \text{if } w_j = 1 \\ \bar{z}_j & \text{if } w_j = 0 \end{cases}$$

- Under this transformation, H_P becomes:

$$H_P = H_{12}^{\text{agree}} + H_{23}^{\text{agree}} + \cdots + H_{n1}^{\text{agree}}$$

Initial Hamiltonian H_B

- We choose the initial Hamiltonian H_B as:

$$H_B = \sum_{j=1}^{n} \left(1 - \sigma_j^x\right)$$

- H_B is invariant under the transformation described earlier.
- The ground state of H_B is:

$$|x = 0\rangle = \frac{1}{2^{n/2}} \sum_{z_1, z_2, \ldots, z_n} |z_1 z_2 \ldots z_n\rangle$$

Adiabatic Evolution Hamiltonian

- The adiabatic evolution Hamiltonian is given by:

$$H(s) = (1-s)\sum_{j=1}^{n}(1-\sigma_j^x) + s\sum_{j=1}^{n}\frac{1}{2}(1-\sigma_j^z\sigma_{j+1}^z)$$

- This Hamiltonian is diagonalized in the space of symmetric states.

Fermion Operator Transformation

- Define the fermion operators for $j = 1, \dots, n$:

$$b_j = \sigma_1^x \sigma_2^x \dots \sigma_{j-1}^x \sigma_j^-$$

$$b_j^\dagger = \sigma_1^x \sigma_2^x \dots \sigma_{j-1}^x \sigma_j^+$$

- These operators satisfy the anticommutation relations:

$$\{b_j, b_k^\dagger\} = \delta_{jk}, \quad \{b_j, b_k\} = 0$$

Ground State Energy and Gap

- The ground state energy $E_{-}(s)$ is given by:

$$E_{-}(s) = 2 - s - \sqrt{(2-3s)^2 + 4s(1-s)(1-\cos(\pi p/n))}$$

- The minimum gap occurs at $s = 2/3$ and is:

$$g_{\min} \approx \frac{4\pi}{3n}$$

- The required evolution time T scales as $T \sim n^3$.

Conclusion

- Quantum adiabatic evolution successfully solves the 2-SAT problem on a ring.
- The evolution time T scales polynomially with n, specifically $T \sim n^3$.
- This method is generalizable to other simple problems with structured constraints.

Level Repulsion

- The phenomenon of **level repulsion** ensures that in typical systems, level crossings are avoided.
- This behavior is common in more complicated systems.
- For small ϵ, the gap g_{min} prevents crossing, ensuring adiabatic evolution.

§ 10: Adiabatic Quantum Computation & Deutsch's Algorithm

Introduction

- Quantum computation harnesses the principle of superposition in quantum mechanics for enhanced efficiency in problem-solving.
- Deutsch's algorithm is one of the earliest quantum algorithms, determining whether a function $f : \{0, 1\} \rightarrow \{0, 1\}$ is constant or balanced.
- The four possible outcomes for f are:
  - $f(0) = f(1) = 0$ (constant)
  - $f(0) = f(1) = 1$ (constant)
  - $f(0) = 0, f(1) = 1$ (balanced)
  - $f(0) = 1, f(1) = 0$ (balanced)

Adiabatic Quantum Computation

- Adiabatic quantum computation replaces quantum gates with a Hamiltonian that evolves continuously with time.
- The system remains in its ground state throughout the evolution.
- Hamiltonians H_0 and H_1 are defined as:

$$H_0 = I - |\psi_0\rangle\langle\psi_0| \quad \text{and} \quad H_1 = I - |\psi_1\rangle\langle\psi_1|$$

where

$$|\psi_0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

and

$$|\psi_1\rangle = \alpha|0\rangle + \beta|1\rangle$$

Coefficients α and β

- The coefficients α and β are defined as:

$$\alpha = \frac{1}{2}\left|(-1)^{f(0)} + (-1)^{f(1)}\right|$$

$$\beta = \frac{1}{2}\left|(-1)^{f(0)} - (-1)^{f(1)}\right|$$

- If f is constant, $\alpha = 1$ and $\beta = 0$.
- If f is balanced, $\alpha = 0$ and $\beta = 1$.

Time-Dependent Hamiltonian

- The time-dependent Hamiltonian is defined as:

  $$H(t) = (1 - s(t))H_0 + s(t)H_1$$

  where $s(t)$ is a function of time such that $s(0) = 0$ and $s(T) = 1$.
- The system evolves adiabatically from the ground state of H_0 to the ground state of H_1.

Eigenvalues of the Hamiltonian

- The matrix elements of $H(s)$ in the basis $\{|0\rangle, |1\rangle\}$ are:

$$H(s) = \begin{pmatrix} 1/2 + s(\beta - 1/2) & -1/2(1-s) \\ -1/2(1-s) & 1/2 + s(\alpha - 1/2) \end{pmatrix}$$

- The corresponding eigenvalues are:

$$E_{\pm}(s) = \frac{1}{2}\left(1 \pm \sqrt{1 - 2s + 2s^2}\right)$$

Evolution Time Estimate

- The minimum gap occurs at $s = 1/2$, and the minimum evolution time T is bounded by:
$$T \geq \frac{1}{\epsilon}$$
where ϵ is the desired accuracy.
- For an accuracy of 90%, the minimum time is approximately $T \approx 4.4$.

Example - Grover's Search

Introduction to Adiabatic Grover Algorithm

- Similar to the circuit model Grover algorithm, the objective is to find the marked item in an unsorted database of N items with the fewest queries.
- Formally, one is allowed to call a function $f : \{0,1\}^n \to \{0,1\}$, where $N = 2^n$, with the promise that $f(m) = 1$ for the marked item m and $f(x) = 0$ for all $x \neq m$.

Hamiltonian Setup

- The final Hamiltonian is $H_1 = I - |m\rangle\langle m|$, where $|m\rangle$ is the marked state.
- The initial Hamiltonian is $H_0 = I - |\phi\rangle\langle\phi|$, where $|\phi\rangle$ is the uniform superposition state:

$$|\phi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle$$

- The time-dependent Hamiltonian is:

$$H(s) = (1 - A(s))H_0 + A(s)H_1$$

where $A(s) = s$ for a linear schedule.

Evolution in the Two-Dimensional Subspace

- The evolution is restricted to the subspace spanned by $|m\rangle$ and $|m^{\perp}\rangle$, where:

$$|m^{\perp}\rangle = \frac{1}{\sqrt{N-1}} \sum_{i \neq m} |i\rangle$$

- In this subspace, the Hamiltonian can be written as:

$$H(s) = \frac{1}{2} I_{2\times 2} - \frac{\Delta(s)}{2} \begin{pmatrix} \cos\theta(s) & \sin\theta(s) \\ \sin\theta(s) & -\cos\theta(s) \end{pmatrix}$$

where $\Delta(s)$ is the gap and $\theta(s)$ is defined by:

$$\cos\theta(s) = \frac{1-2s}{\Delta(s)}, \quad \sin\theta(s) = \frac{2s\sqrt{N-1}}{\Delta(s)}$$

Eigenvalues and Minimum Gap

- The eigenvalues in the two-dimensional subspace are:

$$\epsilon_0(s) = \frac{1}{2}(1 - \Delta(s)), \quad \epsilon_1(s) = \frac{1}{2}(1 + \Delta(s))$$

- The minimum gap occurs at $s = \frac{1}{2}$ and scales as:

$$\Delta_{\min} = \frac{1}{\sqrt{N}} = 2^{-n/2}$$

Adiabatic Condition

- The adiabatic condition requires that the total runtime t_f satisfies:

$$t_f \gg 2 \max_s \frac{\|\dot{H}(s)\|}{\Delta^2(s)}$$

- For the Grover problem, this results in a runtime scaling as $t_f \sim O(\sqrt{N})$, matching the circuit model Grover's algorithm.

Multiple Marked States

- The results generalize to the case where there are $M \geq 1$ marked states.
- The final Hamiltonian is:

$$H_1 = I - \sum_{m \in M} |m\rangle\langle m|$$

- The system evolves in an $M + 1$ dimensional subspace.
- The minimum gap is:

$$\Delta(s) = 1 - 2s^2 + \frac{4M}{N} s(1 - s)$$

§ 11: Quantum Annealing

Quantum Annealing: Introduction

- Quantum Annealing is a quantum algorithm designed to solve optimization problems by evolving a quantum system towards its ground state.
- It is especially useful for solving combinatorial optimization problems.
- Compares closely with simulated annealing but leverages quantum superposition and tunneling to escape local minima more efficiently.

How Quantum Annealing Works (vs. Simulated Annealing)

- Simulated Annealing:
  - Classical algorithm that mimics the annealing process in physics to find the global minimum.
  - Uses thermal fluctuations to escape local minima.
- Quantum Annealing:
  - Uses quantum tunneling to move through energy barriers, potentially escaping local minima more efficiently.
  - Explores many solutions simultaneously due to superposition.
- Advantage: Quantum tunneling allows exploration of regions that classical algorithms may not reach efficiently.

Adiabatic Evolution in Quantum Annealing

- The core principle behind quantum annealing is adiabatic evolution, ensuring that the system reaches the minimum energy state.

Real-World Examples of Quantum Annealing

- D-Wave Systems: The most well-known quantum annealer, used to solve optimization problems in various industries.
- Applications:
  - Logistics: Vehicle routing, supply chain optimization.
  - Finance: Portfolio optimization.
  - Artificial Intelligence: Feature selection and machine learning model training.
- D-Wave has demonstrated significant performance in specific types of optimization tasks.

Thank You!