

# What is Quantum Computing?

- Particles on a very small scale (e.g. electrons), do not follow classical mechanics. Instead, they show a different set of laws, which we call quantum mechanics.
- Quantum computing tries to represent data using quantum properties, and those laws, and perform (quantum) operations on this data.
- For e.g., superposition and entanglement are two quantum properties exhibited at the small scale only.

# Basic Terms

## Classical Bit.

- Take a regular bit  $b$ .
- $b$  can only have a value of 0 or 1. i.e., it is present in exactly 1 of 2 states.
- Hence  $b = 0$  **OR**  $b = 1$ .

## Qubit

- However, a qubit  $q$  is like a vector on the unit circle.

$$q = \alpha i + \beta j$$

- It is present in both states 0 and 1.
- Probability of being present in state 0 =  $\alpha^2$ .
- Probability of being present in state 1 =  $\beta^2$ .
- In other words, a qubit is a combination of the states 0 and 1. It can have any values for  $\alpha$  and  $\beta$  such that  $\alpha^2 + \beta^2 = 1$ .

# Basic Terms

## Classical 2-bit system.

- Take a variable  $X$ , which can be formed from 2-bits.
- Hence,  $X = a_1 a_2$ .
- At any point of time,  $X$  only has 1 of  $2^2$  possible values.
- e.g.  $X = 01$ . (it could also be 00, 10 or 11).

## 2-Qubit System.

- In a 2-qubit system, the value of the qubit, is a combination of **ALL** 4 states.
- $X = a |00\rangle + b |01\rangle + c |10\rangle + d |11\rangle$ .
- Hence, think of it like, some part of  $X$  is present in 00, some part in 01 and so on.
- Here, the probability of  $X$  being in state 00 is  $a^2$ , of being in state 01 is  $b^2$  and so on.
- $a^2 + b^2 + c^2 + d^2 = 1$ .
- This is similar to  $X$  being a 4D unit vector.

## Superposition

- This concept of a qubit being present in all 4 states, with a certain probability of being in each state, is called superposition.

# Representation of a Qubit

- Since a qubit exists in multiple states, it is represented as follows.

$$|\psi\rangle = a|0\rangle + b|1\rangle = \begin{bmatrix} a \\ b \end{bmatrix}$$

- Here,  $|0\rangle$  and  $|1\rangle$  are the orthonormal basis. They are defined in matrix form as:

$$\left\{ |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$$

- Similarly, for a n-qubit system, there are  $2^n$  orthonormal basis. For e.g., for a 2-qubit system, these are

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

# Quantum Gates vs Classical Gates

- **Classical Not gate**

NOT gate	
A	$\bar{A}$
0	1
1	0

- **Quantum Gates**
- One gate is called the Pauli-X gate. It is the equivalent of a NOT gate.

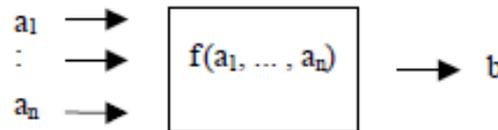
$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

- For e.g., say a qubit  $|\psi\rangle = a|0\rangle + b|1\rangle = \begin{bmatrix} a \\ b \end{bmatrix}$

- Then,  $P * |\psi\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} b \\ a \end{bmatrix}$

# More Quantum Gates (universal)

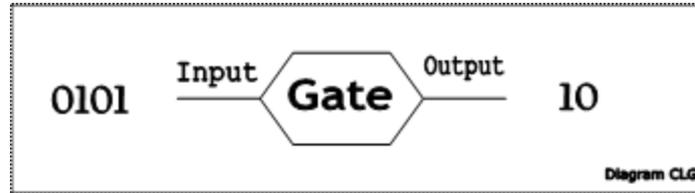
- The three basic classical gates, AND, OR and NOT can be used to solve any classical function of the form



- Hence, together they can be combined to do universal classical computation.
- A NAND gate can be used to simulate all the three gates, and hence it can also be used to do universal computation.
- Similar to this, there is a gate, called the Toffoli gate, which is reversible. Quantum computers can only simulate reversible gates. Since a quantum computer can execute the toffoli gate, it can also be used to do universal classical computation.

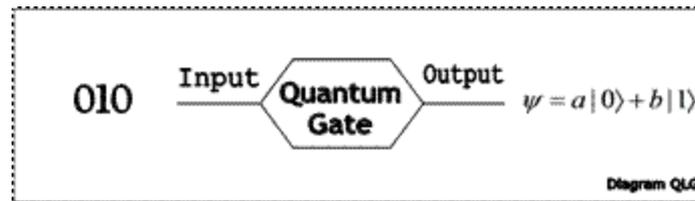
# How to evaluate a function $f$

## Classical Computation



- A classical gate takes input as 0101, and gives output as 10.

## Quantum Computation



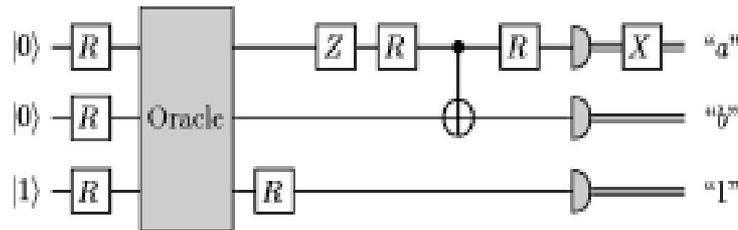
- When a quantum gate takes input (say 010), it gives the output as a superposition of multiple states, and not a single state.

# Why is this helpful?

- Since there is a gate, (Toffoli Gate) which is universal, a quantum computer can do every computation that a classical computer can do.
- However, when using a quantum gate, we get the output as a superposition of multiple states, and not a single state. This is exploited to achieve faster results, than what we can achieve by using a classical gate.

# Quantum Oracle

- In our program, we will also use what is called the quantum oracle.
- An oracle is the portion of an algorithm which can be regarded as a “black box” whose behavior can be relied upon



- Theoretically, its implementation does not need to be specified
- However, in practice, the implementation must be considered

# Quantum Oracle...

- Why do we use oracles?
  - Conceptually simplifies algorithms
  - An oracle hides the details of the implementation, and allows us to focus on the algorithm.
- An oracle can be made up of quantum gates, or it can be made up of classical gates. An oracle, given any input  $X$ , gives us the output  $f(X)$ .

# Measuring a quantum state.

- When we try to measure the value of a quantum state, it collapses to a single basis, just like a regular classical bit.
- A quantum algorithm with classical inputs has to find a way to evolve them into near-classical outputs again for efficient read-out, even though the intermediate state of the system will be decidedly unclassical.

DEALING INTEGER  
PROGRAMS WITH  
ADIABATIC QUANTUM  
COMPUTING

# Outcome of this Research Project

- We develop model of adiabatic quantum computing. We can simulate small adiabatic quantum computer on MATLAB.
- Our model of adiabatic quantum computing produces results that are in accordance with research papers describing state of the art research in Adiabatic quantum computing.
- We are able to describe general mechanism of solving integer programs using adiabatic quantum computer.
- We are able to show that adiabatic quantum computer can solve optimization problem in **constant time** adiabatic evolution.

# Schrödinger's Equation

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

- $\hbar$  stands for Planck's constant
- Operator  $H$  stands for Hamiltonian.
- What does Hamiltonian do?
- Hamiltonian describes energy contents of the system. How?

# Schrödinger's Equation

- How to solve Schrödinger's equation?
- The solution of Schrödinger's equation is

$$|\psi(t)\rangle = \exp\left(-\frac{iHt}{\hbar}\right)|\psi(0)\rangle$$

- Evolution of one quantum state to another is Unitary.
- Possible when Hamiltonian is a Hermitian Matrix.

# Hamiltonian

- Hamiltonian is an operator which we represent by a Hermitian matrix.
- Eigen values of Hamiltonian represent spectrum of possible energy levels (states) of a quantum system.
- The eigenvector (eigenstate) associated with is the lowest eigenvalue energy is the ground state of the Hamiltonian.
- What is the lowest eigenstate (ground state) of this Hamiltonian?

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

$$X = [1 \quad 0 \quad 0]'$$

# Classical Optimization in terms of Quantum states

Given:  $f: \{0,1\}^n \rightarrow \mathbb{N}$ ,  $f(x)$  for  $x = x_1, \dots, x_n$ ,

Objective: find  $x_{\min}$  which minimizes  $f$

$$H = \begin{bmatrix} f(x_{000}) & & & \\ & \cdot & & \\ & & \cdot & \\ & & & \cdot & \\ & & & & f(x_{111}) \end{bmatrix}$$

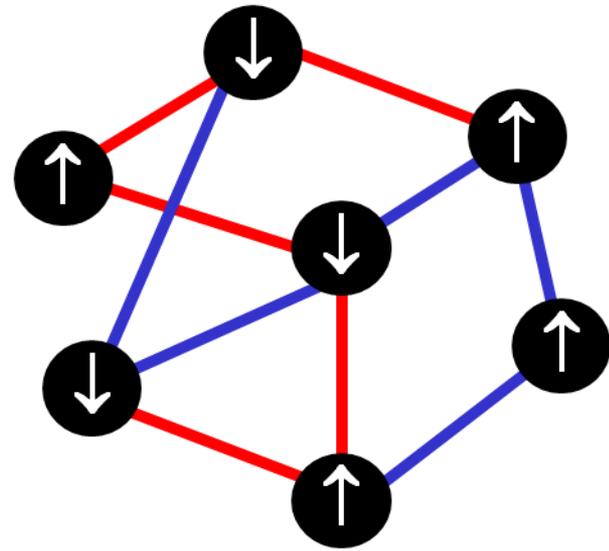
$|x\rangle$  are the eigenvectors

$f(x)$  are the eigenvalues

The answer = state with minimal eigenvalue

# Ground state solution

- Which  $\uparrow\downarrow$  spin distribution maximizes the number of red edges?
- Analogous to a combinatorial optimization problem.
- Lowest energy question for magnetic materials.



Courtesy Dorit Aharonov  
UC Berkeley

The **ground state** of the magnet is solution to our optimization problem!

# Adiabatic Evolution

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

Adiabatic theorem: [BornFock '28, Kato '51]

$$H(0) \longrightarrow H(T)$$

$|\psi(0)\rangle$  Ground state of  $H(0)$   $\longrightarrow$   $|\psi(T)\rangle$  Ground state of  $H(T)$

$$T \gg \frac{1}{\min_s \{\gamma(t)\}^2}$$

$$\gamma(t) = E_1(t) - E_0(t)$$

# Let us start our project

- Imagine there are  $m$  bidders:

$$B_1, B_2, B_3, \dots, B_m$$

- Suppose there are  $n$  items

$$I_1, I_2, I_3, \dots, I_n$$

- The auctioneer will accept bids that maximizes his payoff.

# Quantum Auctions

- An auctioneer gives  $p$  qubits to each bidder.
- The initial state of all qubits is  $|000\dots 0\rangle$
- We can parse our quantum register  $|x\rangle$  as follows

$$|x\rangle = |\text{Item\#}, \text{bidder}_1\_bid ; \text{Item\#}, \text{bidder}_2\_bid ; \dots ; \text{Item\#}, \text{bidder}_m\_bid\rangle$$

# Quantum Auctions

- Let us mention some rules
  - Bidders will prepare their respective qubits and hand them over to the auctioneer.
  - Auctioneer cannot assign same item to multiple bidders.
  - Such an assignment will be infeasible.
  - Auctioneer will select payoffs from available feasible quantum states that we describe next.

# Example of Superposition of Bids

- Suppose we have two bidder B1 and B2 and 1 item
- Bidder1 puts \$2 on the item while Bidder2 puts \$3 on the item
- The resulting state of  $|x\rangle$  will be

$$U_1|00\rangle \rightarrow \frac{|00\rangle + |10\rangle}{\sqrt{2}} \text{ for Bidder1}$$

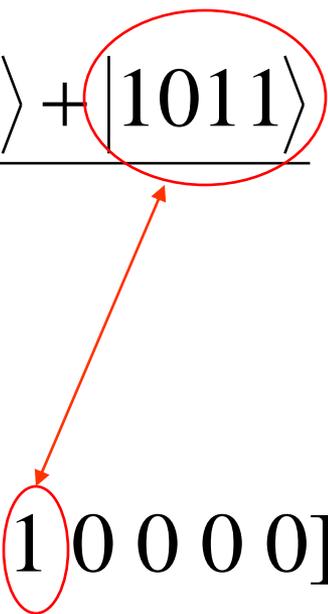
$$U_2|00\rangle \rightarrow \frac{|00\rangle + |11\rangle}{\sqrt{2}} \text{ for Bidder2}$$

# Superposition

- The resulting state of  $|x\rangle$  will be the superposition of qubits prepared by Bidder1 and that by Bidder2. That is,

$$|x\rangle = \frac{|0000\rangle + |0011\rangle + |1000\rangle + |1011\rangle}{2}$$

Infeasible



- In vector form

$$|x\rangle = \frac{1}{2}[1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T$$

# Preparing Hamiltonians

- Initial Hamiltonian  $W$  will contain entries corresponding to the number of ones in each quantum state.

$$W = \text{diag}\{0,1,1,2,1,2,2,3,1,2,2,3,2,3,3,4\}$$

- Final Hamiltonian  $H_f$  will contain diagonal entries corresponding to -ve of payoffs for each quantum bid state.

$$H_f = -\text{diag}\{0,1,2,3,1,0,0,0,2,0,0,0,3,0,0,0\}$$

# Preparing Hamiltonian

- We want  $|x\rangle$  to be the ground state of  $W$ .
- Is this really the case?
- No!
- So we modify our initial Hamiltonian as

$$H_i = UWU^\dagger$$

$$U = U_1 \otimes U_2$$

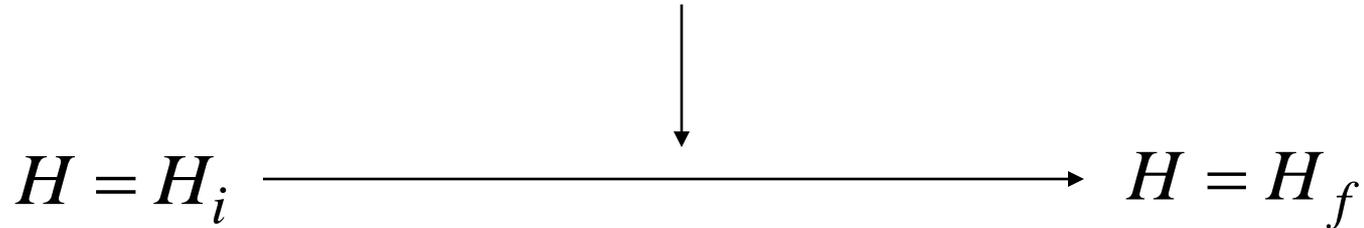
- So that now  $|x\rangle$  is the lowest eigenstate of  $H_i$

# Quantum Auction Protocol

- Start with  $|x\rangle$  the ground state of  $H_i$ .
- Change  $H_i$  slowly so that it becomes  $H_f$
- When Hamiltonian is changing  $|x\rangle$  is also changing.
- The final state  $|x\rangle$  will be the ground state of  $H_f$  which encodes our solution.
- Auctioneer will measure the final state and announce the winner.

# QUANTUM AUCTIONS PROTOCOL

$$H(s) = (1-s)H_i + sH_f$$



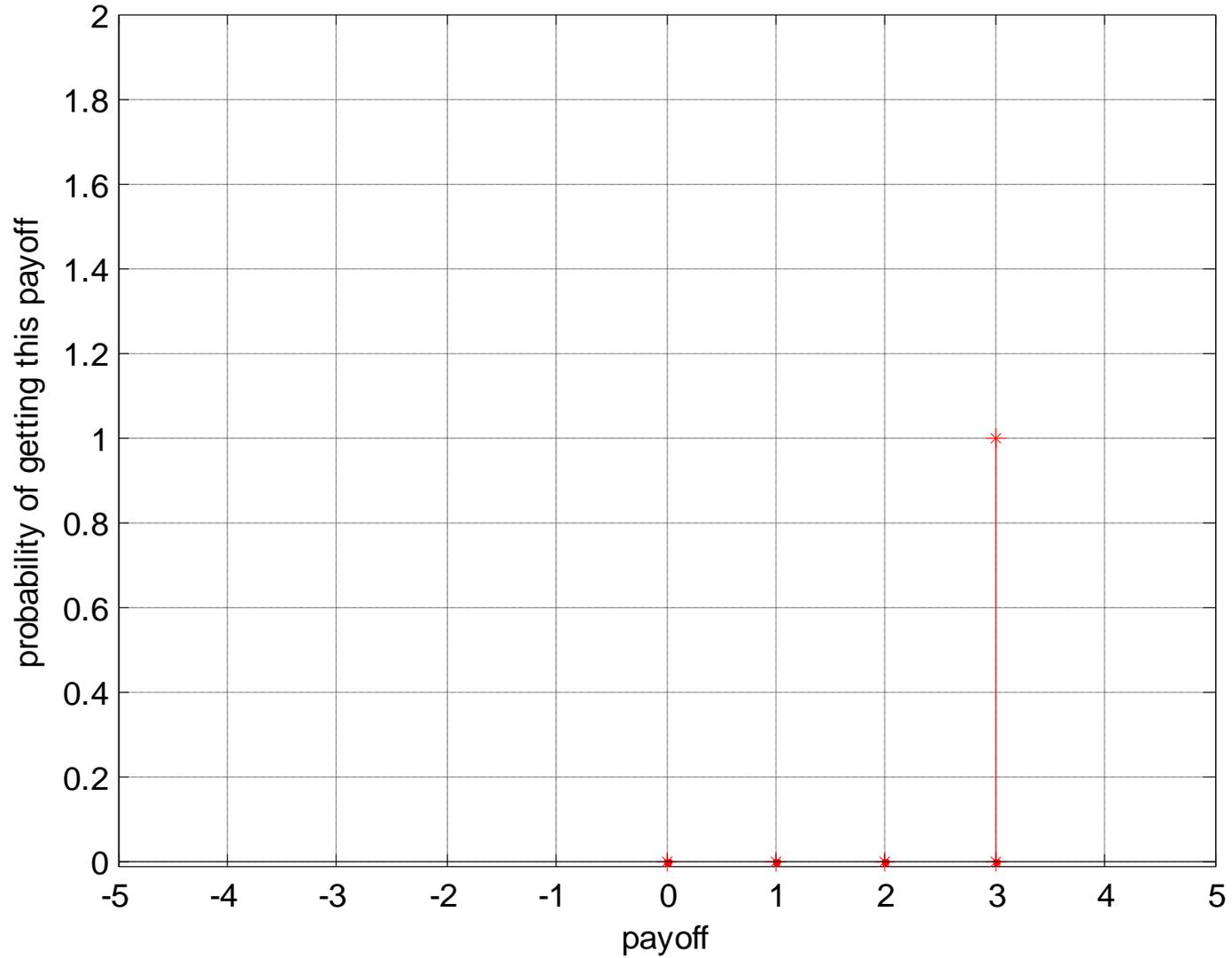
$|x\rangle$  Ground state of  $H(0)$   $\longrightarrow$   $|x\rangle$  Ground state of  $H(f)$

Solution

$$T \gg \frac{1}{\min_s \{\gamma(t)\}^2}$$

$$\gamma(t) = E_1(t) - E_0(t)$$

# DEMO OUTPUT



# DEMO

LET US SEE MATLAB  
DEMONSTRATION OF QUANTUM  
AUCTION PROTOCOL

# Quantum Computer and Integer Programming

- Previous example gives insight into quantum computer solving small integer program namely: Auction winner determination problem.
- Can we extend this approach to solve integer programs in general???

# Quantum Computer and Integer Programming

- Good News
  - It turns out that we can use quantum computer solve integer programs using Adiabatic quantum computing. How?
  - Can we generalize the procedure we followed in the last example.

# Quantum Computer and Integer Programming

- Prepare  $H_f$  the final Hamiltonian which encodes our solution. Can we do this efficiently?
  - For  $2^d$  possible values of input variables, we can calculate corresponding objective values using single black box query!
  - If this is done using NMR Quantum information processor, the corresponding  $2^d$  objective values appear as frequency peaks on NMR spectrometer.
  - These frequencies correspond to eigen values since Energy  $\sim$  frequency

# Quantum Computer and Integer Programming

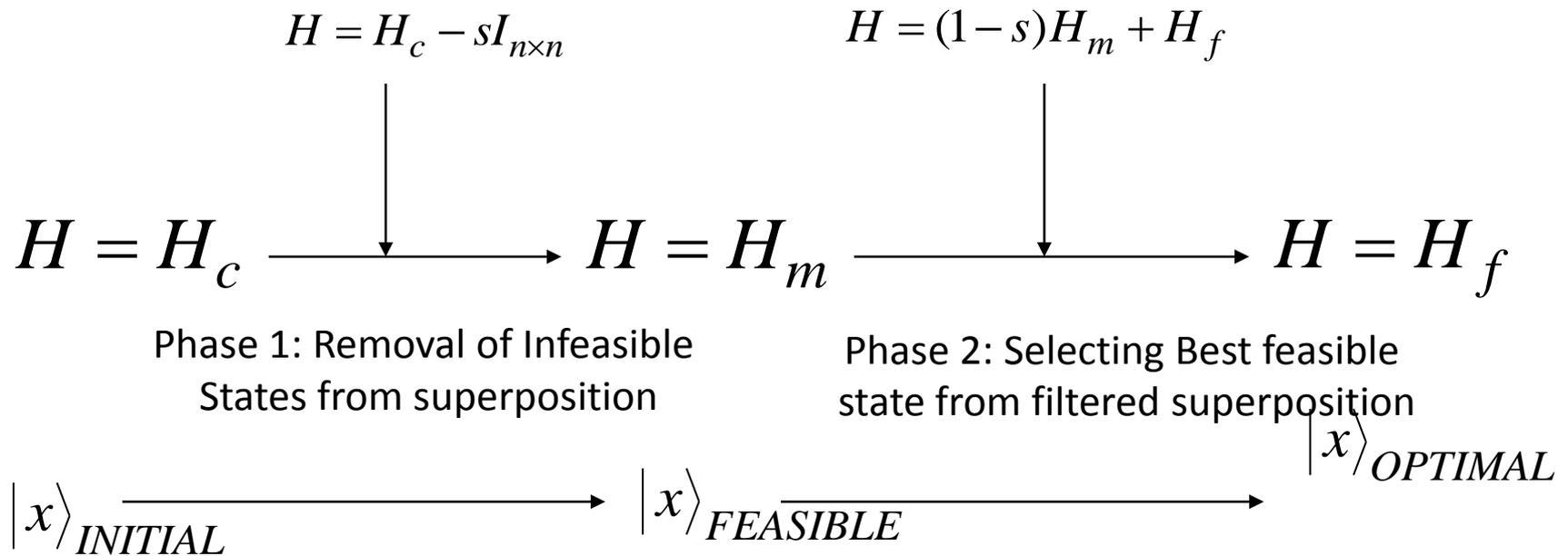
- Prepare  $H_c$  the constraint Hamiltonian containing 1, -1 entries. The  $i$ th entry of this Hamiltonian is 1 if corresponding combination of variable values which is infeasible. How to do this
  - Out of  $2^d$  possible values of input variables, we can determine what combinations of variable values are infeasible using single query to quantum black box!
  - If this is done using NMR Quantum information processor, the mixture of feasible and infeasible states appear as frequency peaks with corresponding phase shifted (0 or 180) on NMR spectrometer.

# Quantum Computer and Integer Programming

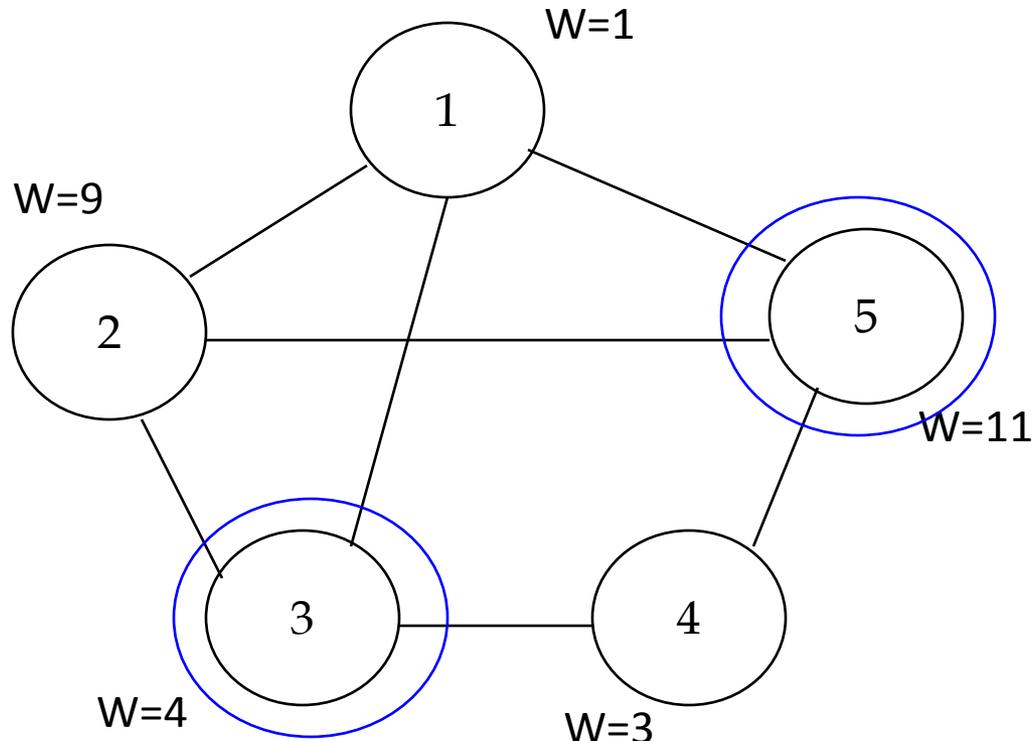
- Prepare  $|x\rangle$  using superposition of all input variable values.  $|x\rangle$  is the initial state of our system.
- Evolve  $H_c$  to  $H_m$ . This evolution is simple. The intermediate Hamiltonian  $H_m$  has eigen values same as those of  $H_c$  but only incremented by 1.
- Evolve  $H_m$  to  $H_f$  and measure the final state which encodes our solution.

# Quantum Computer and Integer Programming

- Adiabatic Quantum Computer solving Integer Program



# Let us Solve Maximum Weight Independent Set Problem using QC

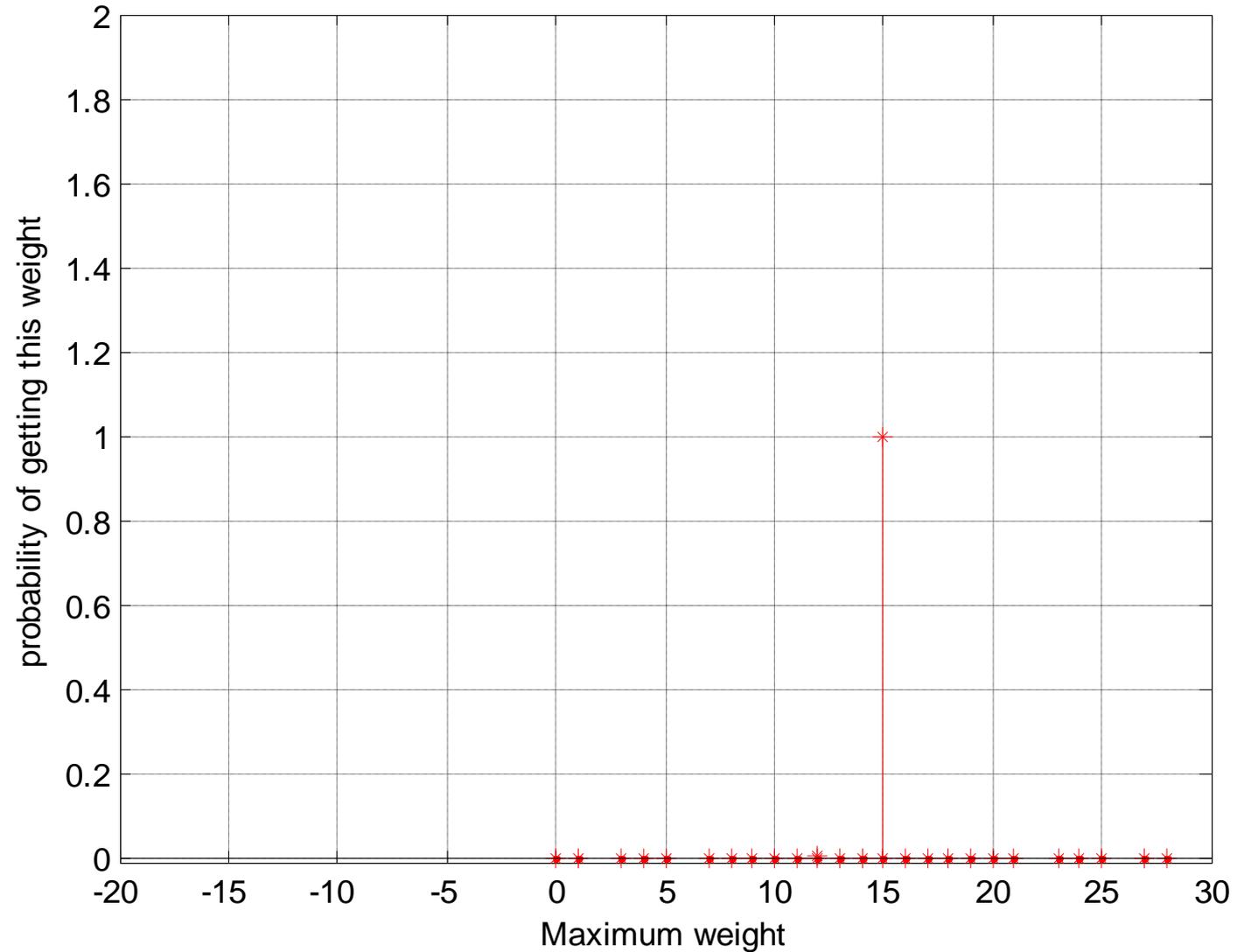


Maximum Weight of Independent vertices is : 15

# DEMO

- Let us see MATLAB demonstration of Adiabatic quantum computer solving Maximum weight independent set of the graph.

# DEMO OUTPUT



# How efficient is Adiabatic Quantum Computation

- The good news is that time spent in the evolution of Hamiltonian doesn't depend upon the size of Hamiltonian!!!
- Time depends on the difference between two lowest eigenvalues also known as **spectral gap**.

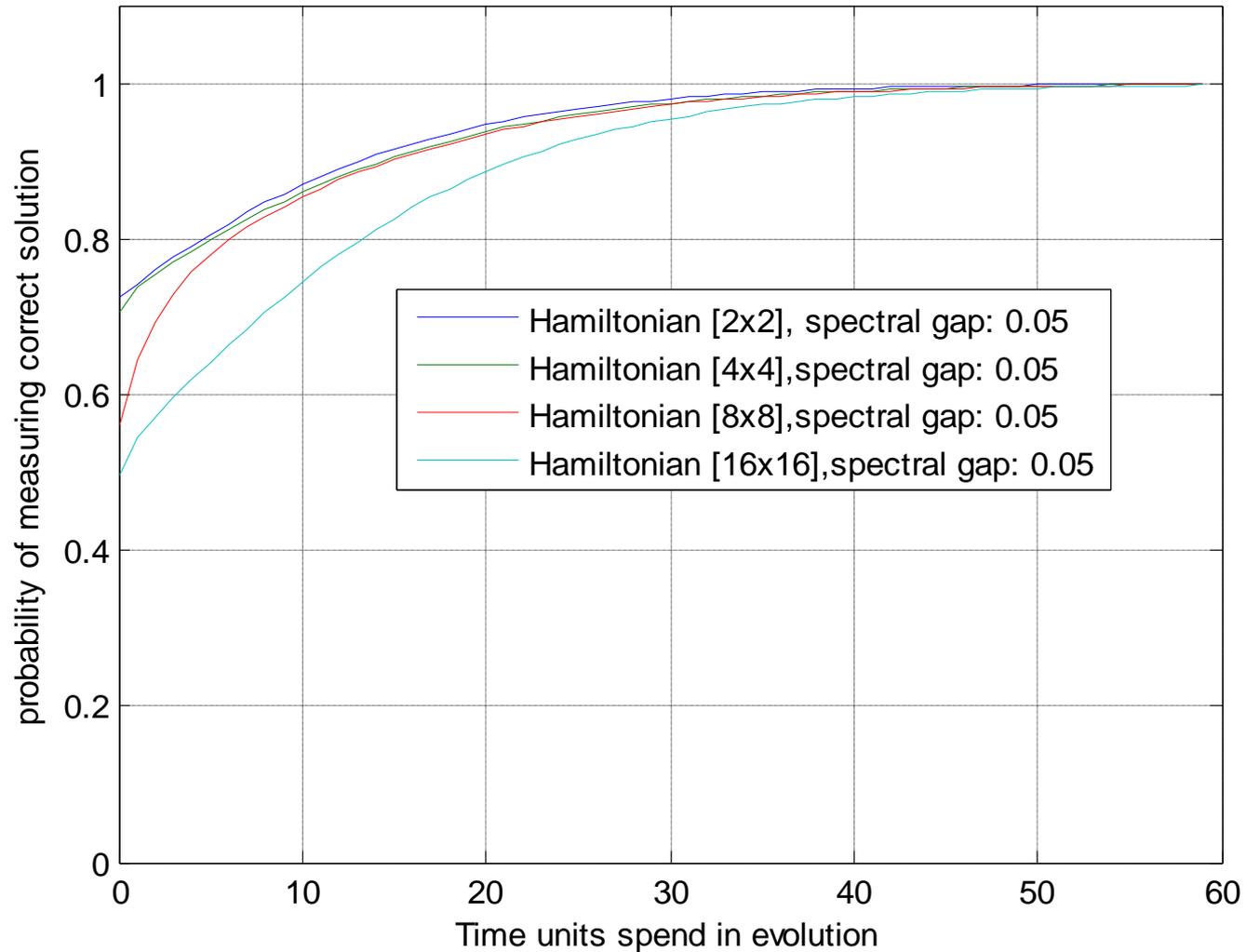
$$T \gg \frac{1}{\min_s \{\gamma(t)\}^2}$$

$$\gamma(t) = E_1(t) - E_0(t)$$

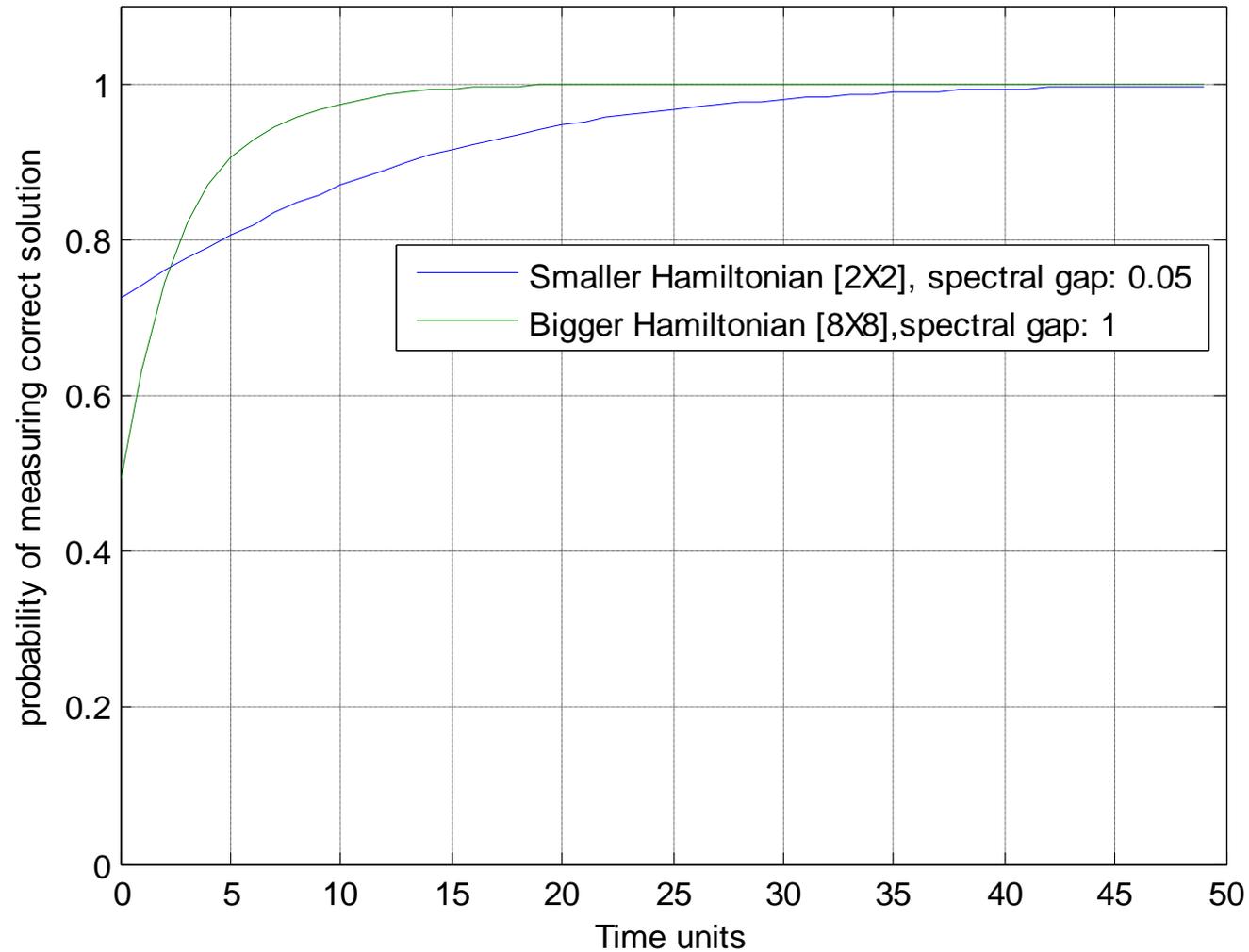
# How efficient is Adiabatic Quantum Computation

- Time spent in evolution depends **ONLY** on difference between two smallest normalized Eigenvalues of evolving Hamiltonian.
- Time spent in evolution **DOESNOT** depend on the number of qubits i.e. size of the problem as such.
- Can we show this?

# How efficient is Adiabatic Quantum Computation



# Efficiency of Hamiltonian



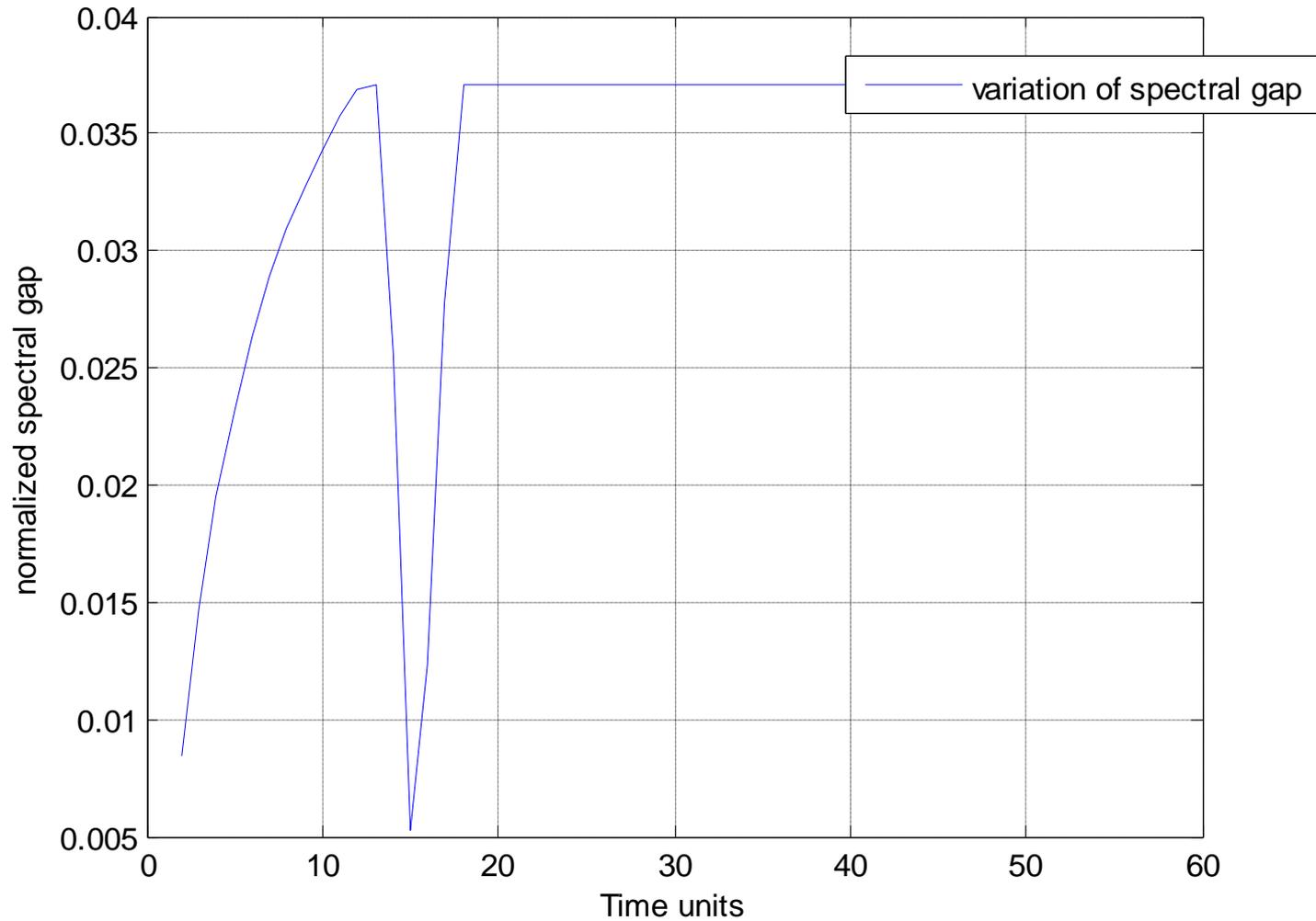
# Time Complexity of our Adiabatic Quantum Computing Algorithm

- **Bad News:**
  - As we increase size of the problem the normalized spectral gap can decrease exponentially in terms of number of qubits!
- **Good News:**
  - We can nullify the exponential decrease in the spectral gap by exponentially increasing the eigenvalues!

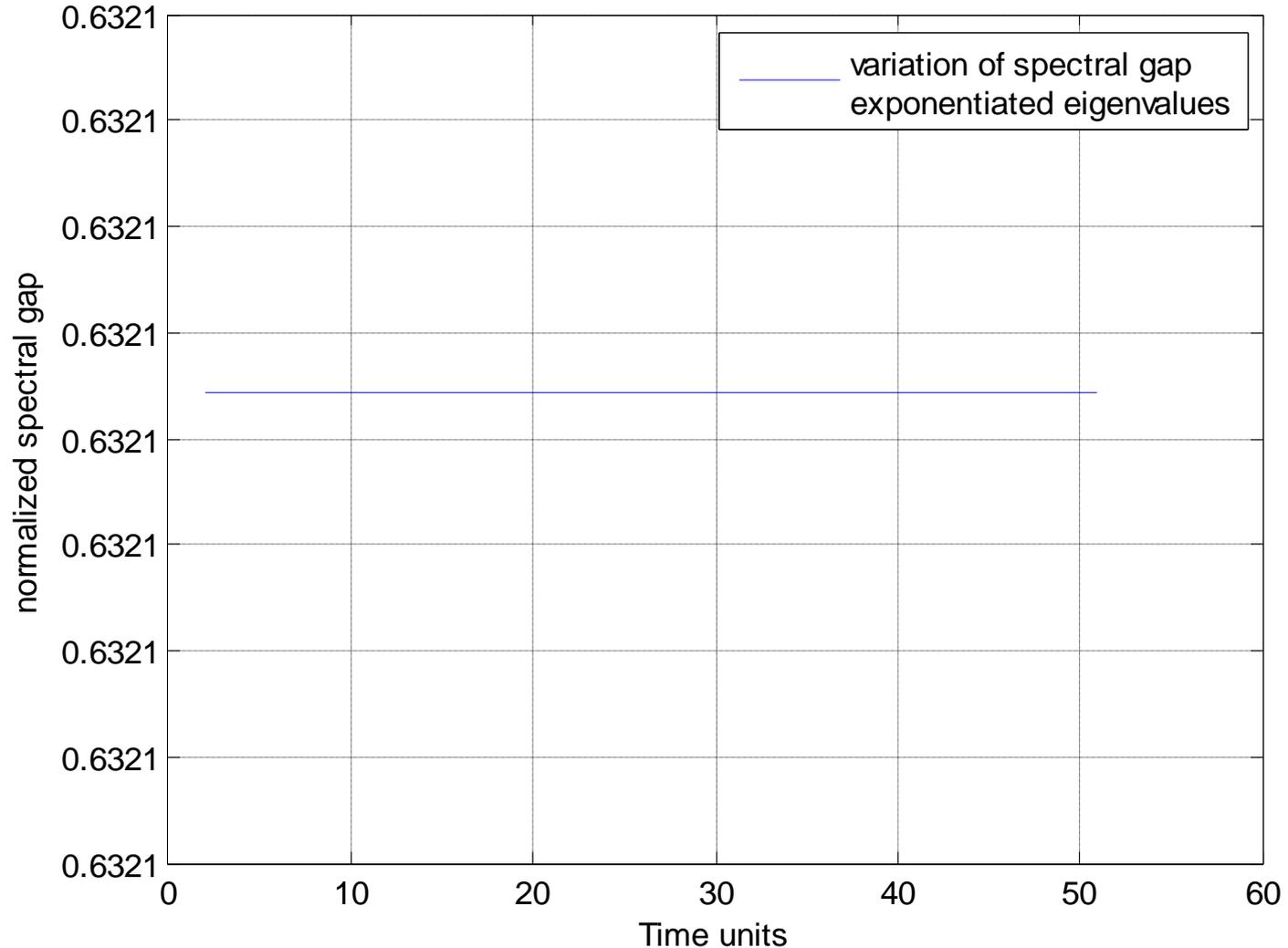
# Time Complexity of our Adiabatic Quantum Computing Algorithm

- In our Maximum weight problem, the eigenvalues of final Hamiltonian may not be simply the sum of feasible combinations of the weights of vertices.
- We can set eigenvalues to be the exponent of sum of feasible weights.
- The resulting quantum algorithm is implemented in `LIP_Final2_mod.m`

# Variation in spectral Gap



# Variation in spectral gap with exponentiated eigenvalues



# How to test our simulation

- Please remember you will be simulating a quantum computer on classical computer.
- How to measure time spent in evolution?
- So we have fixed time step of  $1/50$ . (You can decrease that if you want to)
  - Add more vertices by adding more weights to the weight vector.
  - The output should contain only 1 long stem regardless of how many vertices you add.
  - If this is not the case then adiabatic evolution wasn't completed in constant time. In such a situation please drop me an e-mail at [ahsan@cs.duke.edu](mailto:ahsan@cs.duke.edu)

# Time Complexity of our Adiabatic Quantum Computing Algorithm

- Adiabatic Quantum computing **CAN** solve problem hard optimization problems such that the evolution of Hamiltonians takes **CONSTANT TIME**.
- This idea is not new since Andercut and Ali 2004 showed similar result for searching in item in unstructured database.

# Important questions and Future Work

- How efficiently can we prepare Hamiltonians.
- Study Non linear evolution of Hamiltonians.
- Attempt to tackle exponential number of constraints in a linear program using quantum computer.
- Build general purpose Quantum computer.

THANK YOU