Chapter 9

Quantum Complexity Theory and Adiabatic Computation

9.1 Defining Quantum Complexity

We are familiar with complexity theory in classical computer science: how quickly can a computer (or Turing machine) solve a given problem? To really talk about the power of quantum computers, we need to have notion of how long it takes a quantum computer to solve problems as well. We need quantum complexity theory.

To review, a quantum circuit implements a unitary operator in a Hilbert space of n-qubits: $\mathbb{C}^{2^n}$. A quantum circuit is given in terms of a collection of gates (e.g. CNOT, Haddamard) from some universal gate set, each of which implements a unitary operator on a constant number (say 2) of the n qubits. The total action of all of these gates can be thought of as a single unitary operator that acts on the n input qubits. Unitarity implies that quantum circuits have the same number of inputs as outputs.

We will define our complexity classes in terms of circuits. Let us start by defining the class P of polynomial time computable decision procedures or languages.

Class P - Polynomial Time

A definition of the class P in terms of circuits is the following:

$L \in P$ iff there is a family $\mathfrak{F} = \{C_n\}_{n \in \mathbb{N}}$ of circuits such that:

- $|C_n| \leq poly(n), \forall n \in \mathbb{N}$
- Uniformity The description of the circuit C_n can be computed in time polynomial in n (by a Turing Machine).
- if $|x| = n$ then $C_n(x) = (c \in L?)$

Class BPP - Bounded Error Probabilistic Polynomial Time

In the 70's it was realized that randomness can sometimes speed up computation. Accordingly the class of efficiently solvable computational problems was expanded to probabilistic polynomial time with small probability of error.

A definition of the class BPP in terms of circuits is the following:

$L \in BPP$ iff there is a family $\mathfrak{F} = \{C_n\}_{n\in\mathbb{N}}$ of circuits such that:

- every circuit C_n has an input x of $|x| = n$ bits and a random input r of $|r| = O(poly(n))$ bits
- $|C_n| \leq poly(n), \forall n \in \mathbb{N}$
- Uniformity The description of the circuit C_n can be computed in time polynomial in n (by a Turing Machine).
- moreover:
  - if $x \in L$ and $|x| = n$ then $Pr[C_n(x, r) = "yes"] \geq 2/3$
  - if $x \notin L$ and $|x| = n$ then $Pr[C_n(x, r) = "no"] \geq 2/3$

Class BQP - Bounded Error Quantum Polynomial Time

A definition of the class BQP in terms of circuits is the following:

$L \in BQP$ iff there is a family $\mathfrak{F} = \{C_n \in SU(n)\}_{n\in\mathbb{N}}$ of quantum circuits (unitary operators) such that:

- every circuit C_n has an input x of $|x| = n$ bits and $m = O(poly(n))$ additional inputs of value $|0>$
- the output of the computation is considered to be the outcome of the measurement on the first output of the circuit
- $|C_n| \leq poly(n), \forall n \in \mathbb{N}$
- Uniformity The description of the circuit C_n can be computed in time polynomial in n (by a Turing Machine).
- moreover:
  - if $x \in L$ and $|x| = n$ then $Pr[measure = 1] \geq 2/3$
  - if $x \notin L$ and $|x| = n$ then $Pr[measure = 0] \geq 2/3$

Reversibility and P $\subseteq$ BQP

The construction from the last lecture showing how to convert any classical circuit with n inputs and m gates into a reversible circuit with $O(n+m)$ inputs and $O(n+m)$ gates shows that $P \subseteq BQP$. This is because any reversible circuit can be implemented as a quantum circuit which has the same behavior when the input is a computational basis state.

9.2 BPP $\subseteq$ BQP

We will show that any circuit in BPP can be simulated in BQP by first generating random qubits and then simulating the corresponding polynomial circuit.

Review: BPP

BPP stands for bounded error probabilistic polynomial time. As an example, consider the language PRIMES consisting of prime numbers. There exists a polynomial size circuit C which takes as input x and some random bits r and outputs 1 for ACCEPT and 0 for REJECT.

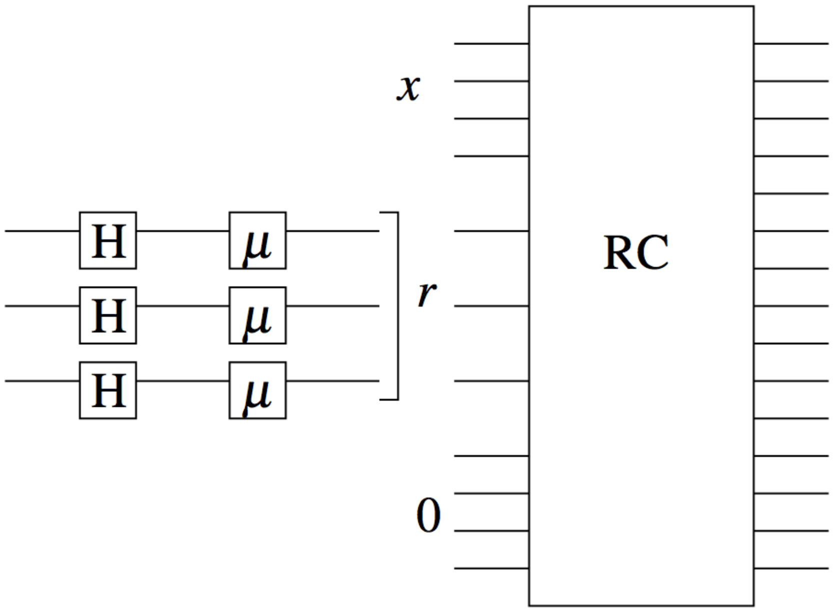
We say PRIMES $\in$ BPP if

$$x \in \text{PRIMES} \Rightarrow \Pr\{C(x,r)=1\} \geq 2/3, x \notin \text{PRIMES} \Rightarrow \Pr\{C(x,r)=0\} \geq 2/3.$$

Simulating BPP

The main difference between a P circuit and a BPP circuit is the additional input of r random bits. We have already shown that any circuit in P can be simulated in BQP. We want to show that it is possible to generate random qubits from $|0\rangle$ inputs. A simple solution is to apply the Hadamard gate to each $|0\rangle$ and then measure. The Hadamard gate converts $|0\rangle$ to $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. Measuring will result is either $|0\rangle$ or $|1\rangle$ with equal probability.

If we generate random bits like this and then run the corresponding quantum circuit to C, we get the straightforward circuit below.



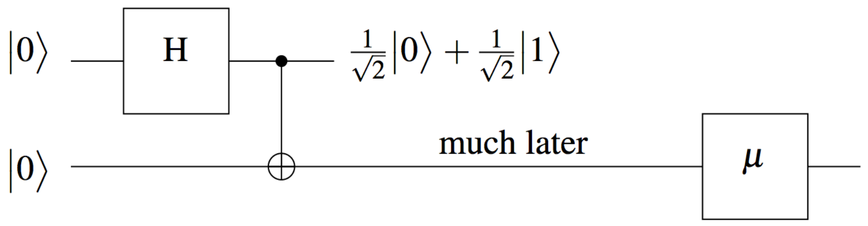
Measurement can be tricky in the intermediate stages of a quantum circuit. Why not skip the measurement and get a superposition of states? Well, if a Hadamard gate occurs in the circuit, we have a problem. The desired outcome is one of these two possibilities with probability 1/2:

$$|0\rangle \longrightarrow \boxed{\text{H}} \longrightarrow \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\ |1\rangle \longrightarrow \boxed{\text{H}} \longrightarrow \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

No interference occurs here. Unfortunately, interference can lead to the following undesirable situation in which the randomness disappears:

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle) \longrightarrow \ \boxed{\text{H}} \ \longrightarrow |0\rangle$$

Measurement prevents quantum interference. But, by the principle of deferred measurement, we can postpone the measurement and get the same result. In fact, we can post the measurement indefinitely and not perform it at all.

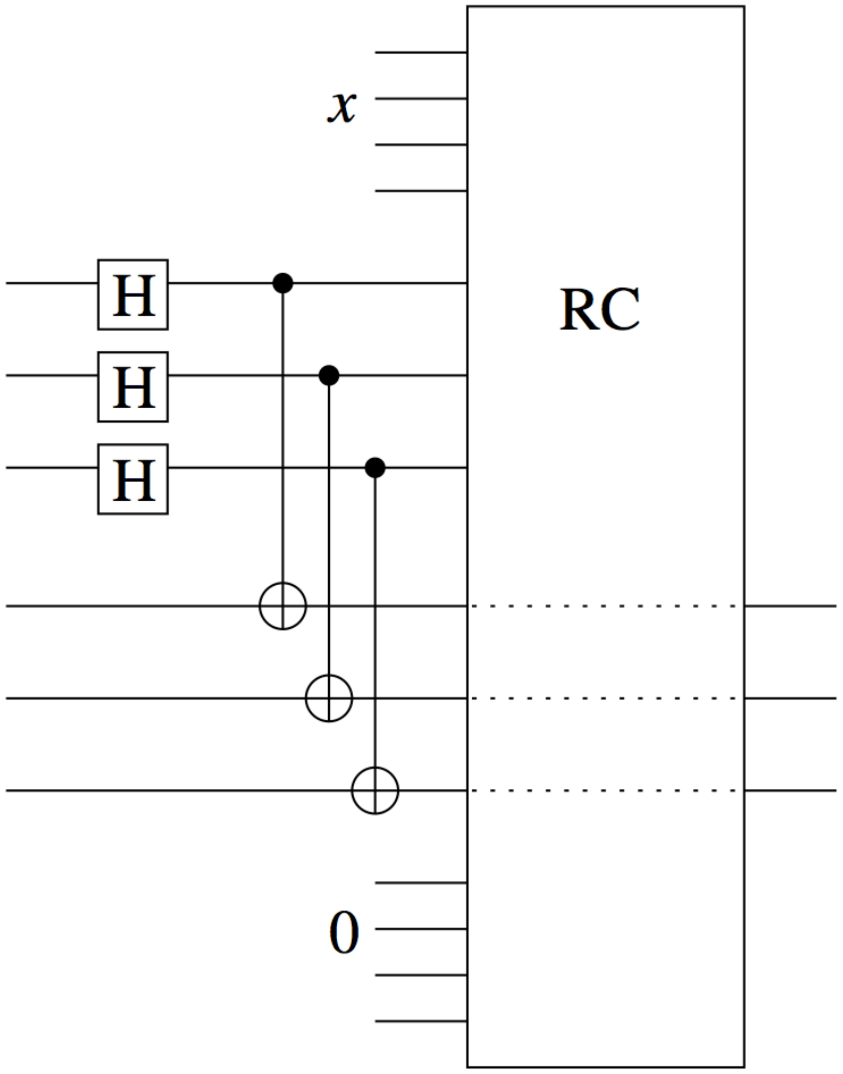


We now need twice as many qubits as before. Half of them are passed through Hadamard gates and connected by CNOT gates to the other half. This fixes the first half of the qubits to either $|0\rangle$ or $|1\rangle$, even though no measurement was made. It is important to note, however, that since the second half of the qubits are now entangled with the first half, we must be certain not to make any measurements on them either.

9.3 $BQP \subseteq PSPACE$

Theorem .2 *$P \subseteq BPP \subseteq BQP \subseteq P^{\#P} \subseteq PSPACE$.*

We give a sketch of the proof that $BQP \subseteq P^{\#P}$. We assume without loss of generality that all the transition amplitudes specified in the transition function δ are real (exercise). The action of a quantum circuit may be described by a tree, each node is labelled with a computational basis state, i.e. a bit string. The root of the tree corresponds to the input $|x\rangle$ and applying a gate to any node yields a superposition of basis states represented by the children of that node. We label the edge to each child by the corresponding amplitude. Let us assume that the quantum circuit accepts or rejects depending upon whether the first qubit, when measured in the computational basis is 0 or 1. Thus each leaf of the tree is either an accepting or rejecting node depending on whether the first bit of the string labeling it is 0 or 1. The amplitude of a path p from the root to a leaf of the tree, β_p, is just the product of the branching amplitudes along the path, and is computable to within $1/2^j$ in time polynomial in j. Several paths may lead to the same configuration c. Thus the amplitude of c after application of T gates is the following sum over all T



length paths p: $\alpha_c = \sum_{p\ to\ c} \beta_p$. The probability that quantum circuit accepts is $\sum_{accepting\ c} |\alpha_c|^2$. Let $a_p = max(\beta_p, 0)$ and $b_p = max(-\beta_p, 0)$. Then $|\alpha_c|^2$ can be written as $|\alpha_c|^2 = \sum_{p\ to\ c}(a_p - b_p)^2 = \sum_{p\ to\ c} a_p^2 + b_p^2 - \sum_{p,p'\ to\ c} 2a_p b_p$. It follows that the acceptance probability of the quantum circuit can be written as the difference between the two quantities $\sum_{accepting\ c} \sum_{p\ to\ c} a_p^2 + b_p^2$, and $\sum_{accepting\ c} \sum_{p,p'\ to\ c} 2a_p b_{p'}$. Since each of these quantities is easily seen to be in $P^{\#P}$, it follows that $BQP \subseteq P^{\#P}$.

In view of this theorem, we cannot expect to prove that BQP strictly contains BPP without resolving the long standing open question in computational complexity theory, namely, whether or not $P = PSPACE$.

9.4 The Adiabatic Model

A somewhat less discussed model of quantum computation is the Adiabatic model, as opposed to the universal gate model. While it will turn out to be equivalent in power to the circuit model, it is interesting for two reasons. First, some local search algorithms can be expressed very elegantly in it. Second, it is a more natural model from a physicists point of view, and therefore potentially applicable to physical realization of quantum computation.

The adiabatic model differs vastly from the unitary gate model that we have been studying. Instead of progressing qubits through a series of unitary gates to realize a desired outcome, the adiabatic model evolves qubits from their input state to their final state by changing the Hamiltonian that governs them with time.

Recall that the time evolution of a state of a closed quantum system is described by *Shrödinger's equation*:

$$\imath \frac{\mathrm{d}}{\mathrm{d}t} |\psi(t)\rangle = H(t) |\psi(t)\rangle$$

For each t, $H(t)$ is a Hermitian operator known as the Hamiltonian of the system, and the eigenvalues of the Hamiltonian are the energy of the corresponding eigenstates. For an n-qubit system, the Hamiltonian $H(t)$ is a $2^n \times 2^n$ Hermitian matrix, i.e. $H(t) = H(t)^\dagger$. For a Hamiltonian H, we call the eigenvector with the smallest eigenvalue the *ground state* of H. Let us also define the *spectral gap* $\Delta(H)$ to be the difference between the second-smallest and smallest eigenvalues, that is, the difference in energy between the ground state and the first excited state.

An adiabatic computation is specified by two Hamiltonians H_B (the base Hamiltonian) and H_P (the problem Hamiltonian). We prepare an initial state $|\psi(0)\rangle$ equal to the ground state of H_B. We design the H_P such that *the ground state of H_P encodes the solution to our computation.* Thus, to perform a computation, we should choose H_B so that the ground state is easy to prepare.

The computation is carried out by evolving $|\psi(t)\rangle$ by a Hamiltonian that interpolates between H_B and H_P. More precisely, for $s \in [0, 1]$, let

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

The crux of the adiabatic model is that we change from H_B to H_P slowly enough, the ground state of H_B will approximately evolve to the ground state of H_P: the direct consequence of the Adiabatic Theorem.

Theorem: Suppose H_B and H_P have unique ground states, and $|\psi(0)\rangle$ is the ground state of H_{init}. If $\psi(t)$ is evolved according to $H(t/T)$ for

$$T \geq O\left(\frac{\|H_B - H_P\|^2}{\epsilon \min_{s\in[0,1]} \Delta(H(s))^3}\right)$$

then $\psi(T)$ is ϵ-close (in ℓ_2) to the ground state of H_P.

To put it simply, the Adiabatic Theorem says that if we want to evolve the ground state of H_B to a state that is ϵ-close to the ground state of H_P, our evolution must take a time T governed by the above equation. Note that the major limiting factor is the size of the spectral gap throughout the evolution. If the spectral gap ever becomes very small, the evolution must progress very slowly.

9.5 3-SAT

To better understand this different paradigm, we will walk through an algorithm for an adiabatic computation. As proposed by Farhi *et al*, there is an elegant solution to the satisfiabilty problem for the adiabatic model. We will study the example of 3-SAT.

The Problem

The satisfiabilty problem, abreviated SAT, is a classic example of an NP-complete problem, or a problem whose solutions can be verified in polynomial time, but cannot necessarily be solved in polynomial time. As it stands, classical computers cannot solve all instances of SAT in polynomial time.

The satisfiability problem asks: given a set of Boolean statements, is there an arrangemnt of their variables that will make all statements true. For example the statements $A \wedge B$ and $A \vee B$ are satisfiable because they are both true when A is true and B is true. However the statements $A \wedge B$ and $\neg(A \vee B)$ are not satisfiable because there is no arrangement of A and B that can take to make both statments true.

The term k-SAT is used to say that each Boolean clause will use only k variables.

The Base Hamiltonian

Our base Hamiltonian should have a ground state that is easy to manufacture, like the state $|0\rangle^{\otimes n}$ for an n bit system. As our first building block, lets consider

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

where $\sigma_x^{(i)}$ is the x-oriented Pauli spin matrix acting on the i^{th} qubit. Then if spin up corresponds to 0 and spin down corresponds to 1, $H_B^{(i)} \mid \psi_i = x\rangle = x$, $x \in \{0, 1)$ so that the ground state of $H_B^{(i)}$ is $\mid 0\rangle$ with energy 0. Now, with our example of 3-SAT, we want to use these initial building blocks to construct a base Hamiltonian for each logical clause C. If clause C is concerned with bits i_C, j_C, k_C, then associate to it the Hamiltonian $H_B^C = H_B^{(i_C)} + H_B^{(j_C)} + H_B^{(k_C)}$. The ground state H_B^C is now $\mid 0\rangle^{\otimes 3}$ with energy 0. Finally we construct our base Hamiltonian by accounting for each clause.

$$H_B = \sum_C H_B^C$$

As desired, the ground state of H_B is $\mid 0\rangle^{\otimes n}$ with energy 0, where n is the number of bits needed to describe all clauses.

The Problem Hamiltonian

Our goal for constructing the problem Hamiltonian is a Hamiltonian H_P whose ground state has energy 0 and is a string of bits that solves the problem if it is SAT, or a state with energy greater than 0 whose bits are the most optimal solution if it is not SAT. This will be constructed in a similar way to the base Hamiltonian, by first considering one clause and then summing over all clauses to construct the final H_P.

Again suppose clause C is concerned with bits i_C, j_C, k_C, and let

$$h_C(i_C, j_C, k_C) = \begin{cases} 0 & \text{if } i_C\ ,\ j_C\ ,\ k_C \text{ satisfy } C \\ 1 & \text{if not} \end{cases}.$$

Lets call the state of the i^{th} bit $|z_i\rangle$ so that the system is described by $|z_1\rangle\,|z_2\rangle\cdots|z_n\rangle$

Now let

$$H_P^C\,|z_1\rangle\cdots|z_n\rangle = h_C(i_C, j_C, k_C)\,|z_1\rangle\cdots|z_n\rangle\,.$$

Note that if there is a configuration of bits that satisfy C, the ground state of H_P^C has energy 0, and the i_C, j_C, k_C bits are such that they satisfy C. Now we put it all together by writing

$$H_P = \sum_C H_P^C \tag{9.1}$$

as before.

This "weeds out" any states that do not satisfy all clauses by assigning an energy penalty to states that fail to satisfy any clause. Thus the ground state of H_P can only have energy 0 if *all* clauses are satisfied. Supposing satisfiability, the ground state of H_P is then a superposition of all solutions, and a measurment will return one solution. Also, if the problem is not satisfiable, the state with the ground will be the state that comes the closest to satisfying the problem. This beautiful result can clearly be extend beyond 3-SAT to solve any SAT problem, and demonstrates a case of solving an NP complete problem with an adiabatic quantum computer. In fact, many discreet optimization problems can be reduced to satisfiabiltiy problems, making this example very powerful.

While this solution is elegant and exciting I cannot make any claims about the time T to run the algorithm. In fact, it is unclear if the run time is in related to the number of bits involved or to the length of the logical clauses any way , which might prove to be an advantage or disadvantage of this paradigm.

Examples

While describing H_P very little information was given about what these operators might look like. Lets construct the clause XOR on the spin qubits ψ_1 and ψ_2. Recall that XOR is only satisfied when $\psi_1 \neq \psi_2$. Thus, let

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

If $\psi_1 \neq \psi_2$, we have $\sigma_z^1\sigma_z^2 \mid \psi_1\psi_2\rangle = - \mid \psi_1\psi_2\rangle$ so that $H_{XOR} \mid \psi_1\psi_2\rangle = 0 \mid \psi_1\psi_2\rangle$. The ground state of H_{XOR} is a superposition of the form $\alpha \mid 01\rangle + \beta \mid 10\rangle$, and has energy 0.

We have prepared an example of an actual 3-SAT problem on 4 bits. Consider the clauses $C_1 = a \wedge (b \vee c)$, $C_2 = a \to (c \wedge d)$, $C_3 = b \wedge c \to \neg d$. It is quick to check that a solution (in fact the only solution) is $\{1011\}$, but this problem is certainly not trivial to solve. We can construct the problem Hamiltonian as in (1), and adiabatically evolve to it from our base Hamiltonian.

Lets consider the state $\psi = \mid 1111\rangle$. $|\psi\rangle$ is a solution to C_1 and C_2, so that $H_P^{C1}\,|\psi\rangle = 0$ and $H_P^{C2}\,|\psi\rangle = 0$. But because $|\psi\rangle$ is not a solution to C_3 so that $H_P^{C3}\,|\psi\rangle = |\psi\rangle$, the energy of $|\psi\rangle$ is 1, and it is not in the ground state of the problem Hamiltonian.

9.6 Equivalence to Universal Gate Computing (optional)

Quantum computing by adiabatic evolution is equivalent to unitary gate computation in power: anything that a unitary gate quantum computer can do in polynomial time, an adiabatic computer can as well, and vise-versa. What follows is a somewhat technical proof of the equivalence between the two models, and is included for the interested reader. The remainder of the course will not focus on adiabatic algorithms, and this section can be skipped.

While we construct adiabatic algorithms from universal algorithms in a complicated way that might be impractical, the importance of this construction is to show that what can be done with universal gates can also be done with adiabatic evolution.

To prove equivalence we will construct unitary gate algorithms that approximate adiabatic evolution algorithms, and vice versa. While both directions can be shown rigorously, mathematical machinery is necessary that lies outside the scope of this course. This will give an overview that is conceptually sufficient and touches on the essential elements of each proof.

Adiabatic $\rightarrow$ Unitary Gate

To create a unitary gate algorithm from our adiabatic evolution we will discreetize the evolution, and progress our qubits along step by step with unitary gates. Consider the ground state ψ_j of our time dependent Hamiltonian $H(s) = sH_P + (1-s)H_B$ at some time s_j with $0 \leq s_j \leq 1$. We want to approximate $|\psi(s_j)\rangle$ by $U|\psi(0)\rangle$ for some unitary transformation U. Note that the complete adiabatic progression of any Hamiltonian as described above induces a unitary transformation that takes $\psi(0)$ to $\psi(1)$. I will state without proof that if we can bind the difference between two Hamiltonians H and H_δ by $\mid H - H_\delta \mid < \delta$, then the unitary transformations U, $U_\delta$are bound by $\mid U - U_\delta \mid < \sqrt{2T\delta}$, where T is the time it takes to adiabatically evolve from H to H_δ.

With this bound in mind, we can break up the progression from $H(0)$ to $H(1)$ into m discrete steps, and induce m unitary transformations. If $|\psi_j\rangle$ is the state of the system after j steps: the ground state of $H(\frac{j}{m}T) = (\frac{j}{m}T)H_P + (1 - \frac{j}{m}T)H_B$, then to progress from $|\psi_{j-1}\rangle$ to $|\psi_j\rangle$ we will use the unitary transformation $U_j = e^{-\imath(T/m)H(jT/m)}$. This simulates subjecting the qubits to the Hamiltonian $H(jT/m)$ for a time T/m. Stringing it all together, the unitary transform U that aproximates the whole evolution is constructed by concatinating each U_j:

$$U = e^{-\imath(T/m)H(1)} \cdots e^{-\imath(T/m)H(j/m)} \cdots e^{-\imath(T/m)H(1/m)}$$

With these m unitary transformations, we essentially step our initial ground state $\psi(0)$ along to $\psi(T)$. Because of the bound described above, we can calculate how small we need to make our steps.

Unitary Gate $\rightarrow$ Adiabatic

Now we want to show that a problem Hamiltonian can be constructed from some quantum circuit. It is difficult to encode the outcome of a unitary gate transformation exactly as the ground state of our Hamiltonian, but we can encode it as *one* ground state of the problem Hamiltonian. Consider the state

$$|\alpha\rangle = \frac{1}{\sqrt{L+1}} \sum_{l=0}^{L} |\psi_l\rangle \otimes \left|1^l 0^{L-l}\right\rangle_{clock}$$

where $|\psi_l\rangle$ is the state of the circuit after the l^{th} unitary gate, and the qubits $\mid 1^l 0^{L-l}\rangle$ labeled *clock* counts the gate l. If we can make $\mid \alpha\rangle$ the ground state of our Hamiltonian, we can measure until we find the solution state $\mid \psi_L\rangle \otimes \mid 1^L\rangle$ which is clearly marked by the clock qubits. In this way, we can learn the state of the circuit after l gates for any $0 \leq l \leq L$, and we expect to find the final state after only $\sqrt{L}$ measurements.

To construct this Hamiltonian, we will apply energy penalties to states that are not desired and weed them out. The problem Hamiltonian H_p will be composed of three parts, H_{clock} which checks that the clock bits are correct, H_{input} which mandates that the computational qubits are all 0 when $l = 0$, and H_l which checks that the propagation from $\mid \psi_l\rangle$ to $\mid \psi_{l+1}\rangle$ was correct. The Hamiltonians H_{clock} and H_{input} are relatively easy to construct from projection operators, are less essential to understanding the problem, so I will state without proof that they can be constructed and that $\mid \alpha\rangle$ is a ground state as desired. H_l, however, is more difficult to construct. Consider the operator

$$H_l = I \otimes |100\rangle \langle 100|^1_{clock} - \left[U_l \otimes |110\rangle \langle 100|_{clock} + U_l^\dagger \otimes |100\rangle \langle 110|_{clock}\right]^2 + I \otimes |110\rangle \langle 110|^3_{clock}.$$

Note that every component of H_l is a unitary operator with a 3-qubit operator tacked on to the end of it. Though it is not clearly expressed in the above formula, *it is understood that the 3-qubit operators act on the* $l-1$, l, $l+1$ *bits of the clock.* This is very important, as these 3-qubit operators progress the clock forward or backward as in 2, or keep it the same as in 1 or 3.

This Hamiltonian can be understood by examening piece 2, whose purpose is to assign energy penalties to states that do not propagate correctly across the unitary gate U_l. We see that

$$U_l \otimes |110\rangle \langle 100| \left(|\psi_l\rangle \otimes |100\rangle\right) = |\psi_{l+1}\rangle \otimes |110\rangle$$

so that

$$I \otimes |110\rangle \langle 110|^3_{clock} - U_l \otimes |110\rangle \langle 100|_{clock} = 0$$

for states that propagate correctly under U_l. Similarly, $I \otimes |100\rangle \langle 100|^1_{clock} - U_l^\dagger \otimes \mid 100\rangle\langle 110 \mid_{clock}$ checks the reverse direction. We have constructed H_l so that its ground state is $\mid \psi_l\rangle \otimes \mid 100\rangle + \mid \psi_{l+1}\rangle \otimes \mid 110\rangle$.

Putting this all together we see that

$$H_P = \left(\frac{1}{2}\sum_{l=0}^{L} H_l\right) + H_{input} + H_{clock}$$

has the ground state $\mid \alpha\rangle$ as described earlier. The $\frac{1}{2}$ is there to correct for the double counting of each $|psi_l\rangle$. We also see that in the construction of H_P, the number of terms that construct the Hamiltonian grows as a polynomial of the number of gates. This does not allow us to make any claims about the length of time the adiabatic evolution will take, but it is an interesting result none-the-less.

Exercise: Construct the Hamiltonian H_{clock} that ensures that the clock bits are are correct.