Quantum algorithms for optimization

by Sumeet Shirgure

USC, Spring 2022

Abstract

This term paper documents and discusses existing methods and some recent developments in the field of quantum computing in solving combinatorial and optimization problems.

1 Introduction

In recent years there has been considerable progress in quantum computing theory and technology. An interesting sub-field of applications of quantum algorithms is in solving optimization problems faster than classically possible. However, since the information processed by such algorithms is quantum in nature, we must also be aware of their utility and limitations.

This paper attempts to discuss these topics at a high level without going too deep into the technical details, and is supposed to be accessible to anyone without a thorough background in quantum computing. Section 2 gives a bare minimum introduction to quantum mechanics and quantum computation, and only requires a working knowledge of linear algebra. Section 3 introduces some standard tools in quantum computing. And the later sections each discuss a relevent topic in adequate detail. A few topics that serve to tie some loose ends are deferred to the appendices. A reader uninterested in quantum computing might still find the contents of section 6 of considerable interest, because the majority of that discussion is in the classical domain.

2 A crash course in quantum computing

A short introduction to quantum computing is [34]. Here we review some basic concepts at the bare minimum, which are discussed in any text [29] on the subject. We will start by reviewing three basic postulates of quantum mechanics – (a) state description, (b) state collapse and measurement statistics, and (c) evolution.

2.1 Postulates

Quantum information is processed and stored in quantum computers in the form of quantum bits or qubits. The state of any quantum mechanical system is postulated to be a vector ψ belonging to a complex Hilbert space $\mathcal{H}$ that is normalized w.r.t that inner product $\langle .,. \rangle$. Qubits are quantum systems for which this Hilbert space is the 2D complex vector space $\mathbb{C}^2$.

As a mathematical abstraction, every qubit state is given by a normalized 2D complex vector $\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \alpha|0\rangle + \beta|1\rangle, |\alpha|^2 + |\beta|^2 = 1$. $|0\rangle$ and $|1\rangle$ (section 2.3) being a basis of $\mathbb{C}^2$ are also called the *computational basis* states. When parametrizing the qubit state by $e^{i\gamma}(\cos(\theta)|0\rangle + e^{i\varphi}\sin(\theta)|1\rangle)$, we can ignore γ, called the global phase, as we will see it has no *observable effects*.

Physical quantities of interest like position, momenta, energy, spin are are represented by *observables*, which are linear operators acting on $\mathcal{H}$. An important example is the energy operator $\hat{H}$ - the Hamiltonian. A fundamental idea in quantum mechanics is that the eigenvalues of any observable correspond to physically observed quantities. Since these must be real even for complex vector states, the observables are always Hermitian, i.e they must have real eigenvalues : $\hat{H}^\dagger = \hat{H}$.

This possibility of a discrete nature of energy eigenvalues is at the heart of quantum mechanics.

The eigenvectors corresponding to an operator are called its *eigenstates*. If a quantum system $|\psi\rangle$ is in a linear combination of eigenstates $\sum_j \alpha_j|\psi_j\rangle$, it's said to be in a *quantum superposition* of those states. When the associated physical quantity observed, such a state is postulated to *collapse* to the respective eigenstate $|\psi_j\rangle$. The probability of observing m is given by the Born rule : $|\alpha_m|^2/\sum_j |\alpha_j|^2$. That is, the probabilities are proportional to the magnitudes of the corresponding *amplitudes* α_j. This is why the global phases are unobservable. The state $|\psi_j\rangle$ after the collapse is again normalized.

E.g the qubit state $\alpha_0|0\rangle + \alpha_1|1\rangle$ when measured in the computational basis is put in the state $|j\rangle$ after measurement with probability $|\alpha_j|^2$, and we observe j as some physical phenomenon.

Finally, the continuous time evolution of a closed quantum system is described by the Schrödinger equation $i\hbar\frac{d}{dt}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle$, $\hat{H}$ being the energy operator. $\Rightarrow|\psi(t)\rangle = e^{-i\hat{H}t/\hbar}|\psi(0)\rangle$, where the exponentiation is the usual analytic function of the operator $\hat{H}$. The Hamiltonian is *time-independent* since closed systems have conserved energy.

$U(t) = e^{-i\hat{H}t/\hbar}$ is a unitary operator, as it should be to keep ψ normalized –

$$U(t)^\dagger = \sum_{k\geqslant 0}\frac{1}{k!}\left(\left(\frac{-i\hat{H}t}{\hbar}\right)^k\right)^\dagger = \sum_{k\geqslant 0}\frac{1}{k!}\left(\frac{i\hat{H}t}{\hbar}\right)^k = e^{i\hat{H}t/\hbar} = U(t)^{-1}$$

The exponential of a skew-Hermitian operator is always unitary. As we will see, quantum logic gates must also be unitary. This is a key necessary condition to prevent loss of information, which is essential when we consider thermally isolated closed systems – a consequence known as Landauer's principle [26].

2.2 The quantum gate model

The standard model of quantum computation is described in terms of a series of *quantum logic gates* applied on a set of qubits. Each gate acts on a small subset of qubits. Let's start by understanding how to express composite states with multiple qubits.

2.2.1 Composite states

The main idea is to take products of the corresponding Hilbert spaces. The composite state is then the tensor (Kronecker) product of the two states. E.g if two qubits are in states $\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \alpha|0\rangle + \beta|1\rangle$ and $\begin{pmatrix}\mu\\ \nu\end{pmatrix} = \mu|0\rangle + \nu|1\rangle$, the composite system is in the state $\begin{pmatrix}\alpha\\ \beta\end{pmatrix} \otimes \begin{pmatrix}\mu\\ \nu\end{pmatrix} = \begin{pmatrix}\alpha\mu\\ \alpha\nu\\ \beta\mu\\ \beta\nu\end{pmatrix}$.

$=(\alpha|0\rangle + \beta|1\rangle)\otimes(\mu|0\rangle + \nu|1\rangle) = \alpha\mu|0\rangle\otimes|0\rangle + \alpha\nu|0\rangle\otimes|1\rangle + \beta\mu|1\rangle\otimes|0\rangle + \beta\nu|1\rangle\otimes|1\rangle$

$|j\rangle\otimes|k\rangle$ is abbreviated as $|jk\rangle$. Sequences of qubits in computational basis can be represented as bitstrings inside a ket $|\rangle$. The above state is written in the $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ basis of the product Hilbert space $\mathbb{C}^{2^2}$. Hence, n qubits have $\mathbb{C}^{2^n}$ dimensional states. This apparent complexity is not entirely observable, however. Still quantum computers can process information by moving around in such large spaces.

Yet another crucial phenomonon is that of *entangled states*. There exist states like $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$ that can't be expressed as the tensor product of two states. Such entangled states are physically realizable, and imply that measuring one qubit will immediately tell us about the state of the other. Even if this state itself is not a product of two single-qubit states, it does lie in the product Hilbert space of the two qubits, which is what the postulate requires.

Measuring just the first qubit of $\sum_{pq}\alpha_{pq}|pq\rangle$ results in an observation corresponding to the $|0\rangle$ state, the final state is $\frac{\sum_q \alpha_{0q}|0q\rangle}{\sum_q |\alpha_{0q}|^2}$. The probability of this happening is itself $\sum_q |\alpha_{0q}|^2$.

2.2.2 Qubit gates

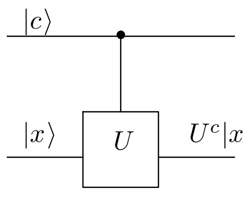
State evolution must be unitary to keep ψ normalized. The states of qubits are manipulated using quantum gates that are unitary operators acting on corresponding Hilbert spaces. E.g, the NOT gate $X = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$ maps $\alpha|0\rangle + \beta|1\rangle$ to $\beta|0\rangle + \alpha|1\rangle$, effectively acting as a negation. Any 2×2 unitary matrix U acts on a qubit in state $|\psi\rangle$ to give $U|\psi\rangle$.

Multiple single-qubit gates U_i, U_j acting on $|\varphi\rangle, |\psi\rangle$ is equivalent to $U_i \otimes U_j$ acting on $|\varphi\rangle \otimes |\psi\rangle$: $U_i \otimes U_j(|\varphi\rangle \otimes |\psi\rangle) = (U_i|\varphi\rangle) \otimes (U_j|\psi\rangle)$. More generally, a single unitary operation U can act on multiple qubits. E.g, the controlled-NOT, or CNOT$=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ mapping $|p\rangle \otimes |q\rangle \rightarrow |p\rangle \otimes |q \oplus p\rangle$ where $\oplus$ is bitwise exclusive or. Note how it's convenient to just talk of the action of quantum gates on basis states, because linearity uniquely extends the definition to superposition states.

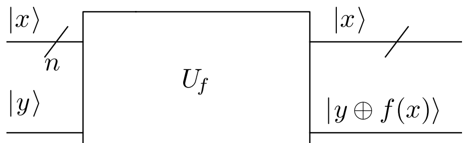
The set of all single qubit unitaries along with CNOT is a universal gate set. And even if the number of unitary transformations is uncountably infinite, there always exist modest-sized finite gate sequences that approximate any desired operation up to an acceptable error in the operator norm – a fact known as the Solovay-Kitaev theorem [17].

Quantum gates are a generalization of classical gates. However, classical gates like AND and OR destroy information irreversibly. Their quantum analogs can be implemented using reversible gates like the Toffoli gate $CCX|x\rangle \otimes |y\rangle \otimes |z\rangle \rightarrow |x\rangle \otimes |y\rangle \otimes |z \oplus (x \wedge y)\rangle$. Bennet [7] showed that any classical computation can be made reversible like this, with an acceptable overhead in the number of qubits and gates.

Some standard qubit gates are the Pauli gates X, $Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$, $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, the Hadamard gate $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$, phase shift gates $P(\varphi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i \varphi} \end{pmatrix}$. A common operation is to put a set of n qubits each in the uniform superposition $(|0\rangle + |1\rangle)/\sqrt{2} = H|0\rangle$. The composite state $\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)^{\otimes n} = 2^{-n/2}\sum_{x \in \mathbb{Z}_2^n} |x\rangle$ is a uniform sum of all bit strings. This can, for example, represent the search space of all 2^n subsets in a search problem.



(a) A controlled unitary



(b) A reversible unitary computing f

Figure 1.

Like the CNOT gate, we may condition any unitary on the state of an external qubit, depicted diagramatically as in figure 1(a). $|c\rangle \oplus |x\rangle \longrightarrow |c\rangle \oplus U^c|x\rangle$; U is applied only when the control qubit is $|1\rangle$. The equivalent unitary is $(|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U)$ (see section 2.3). Controlled unitaries can hence be extended to arbitrary circuits being conditioned on another qubit. Physical implementation of controlled unitaries is an issue we won't be concerned with.

Any n bit circuit $f: \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2$ can be implemented as a unitary operation U_f (figure 1(b)) acting on $n+1$ qubits that acts as $U_f|x\rangle \otimes |y\rangle \longrightarrow |x\rangle \otimes |y \oplus f(x)\rangle$. Applying U_f to $|x\rangle \otimes |0\rangle$ gives us $|x\rangle \otimes |f(x)\rangle$. U_f may require additional $|0\rangle$ qubits as temporary memory, but we can always return that temporary memory to its original state by *uncomputing* the intermediate garbage.

Consider the effect of U_f on $2^{-n/2}\sum_{x \in \mathbb{Z}_2^n} |x\rangle \oplus |0\rangle \longrightarrow 2^{-n/2}\sum_{x \in \mathbb{Z}_2^n} |x\rangle \oplus |f(x)\rangle$. The physical apparatus implementing U_f must somehow simultaneously evaluate f at every point. And yet, we cannot access more than one function output without measuring the resulting state more than once. Even still, such *quantum parallelism* is useful for designing quantum algorithms.

2.2.3 Measurement

Measurement of a qubit collapses its state into a $|0\rangle$ or a $|1\rangle$, which we then observe as a classical bit. Since measurement is non-unitary, it can be used to affect non-unitary transformations that we might need. E.g consider the state $|\psi\rangle = \frac{|00\rangle + |01\rangle + |10\rangle}{\sqrt{3}}$. We cannot prepare this just by using unitary gates on two qubits initialized to $|0\rangle$. We can however add an ancillary qubit, and use it to store the AND of the first two :

$|000\rangle \rightarrow [H^{\otimes 2} \otimes I] \rightarrow \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \otimes |0\rangle \rightarrow [CCX] \rightarrow \frac{|000\rangle + |010\rangle + |100\rangle + |111\rangle}{2}$

Now, we just measure the 3rd qubit, and keep repeating the experiment until we observe it collapse to $|0\rangle$; which would then imply that the first two qubits are in state $|\psi\rangle$. The required observation occurs with probability $3/4$, so we need $4/3$ repetitions of the experiment in expectation. We can hence use partial measurement also as a technique to create quantum states via non-unitary transformations.

2.3 The bra-ket notation

Lastly, since Dirac's bra-ket notation is quite convenient, we'll discuss a bit more about it here. $|x\rangle$ (pronounced *ket*) is used to denote a vector in space V. $\langle x|$ (*bra*) is the complex conjugate transpose of $|x\rangle$, and is a dual vector in the dual space V^*. The standard inner product of two complex vectors $\langle x,y\rangle = (|x\rangle)^\dagger|y\rangle = \sum_i \bar{x_i} y_i$ is denoted as $\langle x|y\rangle$. ($\bar{x_i}$ denoting complex conjugates.) As usual, $\langle x,y\rangle = \overline{\langle y,x\rangle}$. The x inside $|x\rangle$ can be anything as per our convenience.

Bra-ket notations are useful to describe operators as sums of outer products. E.g a unitary U mapping an orthonormal basis e to another e' as $U|e_j\rangle \rightarrow e^{i\varphi_j}|e_j'\rangle$, must be of the form $\sum_j e^{i\varphi_j}|e_j'\rangle\langle e_j|$. Indeed $U|e_k\rangle = (\sum_j e^{i\varphi_j}|e_j'\rangle\langle e_j|)|e_k\rangle = \sum_j e^{i\varphi_j}|e_j'\rangle\langle e_j|e_k\rangle = \sum_j \delta_{kj} e^{i\varphi_j}|e_j'\rangle = e^{i\varphi_k}|e_k'\rangle$, where $\langle e_k|e_j\rangle = \delta_{kj}$ is 1 iff $k=j$ and is 0 otherwise.

As another example, if an observable H has eigenvalues h_i and eigenstates $|u_i\rangle$, the spectral theorem states $H = \sum_i h_i|u_i\rangle\langle u_i|$. Note that $|u_i\rangle\langle u_i|$ is a projection on the i^{th} eigenvector of H.

The expected eigenvalue of an observable $H = \sum_i h_i|u_i\rangle\langle u_i|$ exhibited by a state $|\psi\rangle$ is $\sum_i h_i|\langle u_i|\psi\rangle|^2 = \sum_i h_i\langle\psi|u_i\rangle\langle u_i|\psi\rangle = \langle\psi|(\sum_i h_i|u_i\rangle\langle u_i|)\,|\psi\,\rangle = \langle\psi|H|\psi\rangle$, where $\langle u_i|\psi\rangle$ are the projections of ψ along the eigenstates of H, and the l_2-norm-squares are probabilities by the Born rule.

3 Some standard quantum algorithms

Here we discuss some standard quantum algorithms. These are useful techniques for solving combinatorial problems in themselves, and can also be used to design more sophisticated ones. This is by no means a complete collection.

3.1 Amplitude amplification

Discovered by Grover [23], and independently by Brassard et. al [13] amplitude amplification referst to a broad set of techniques to manipulate superpositions into states having higher amplitudes associated with eigenstates we prefer. As an example we'll look at Grover's algorithm.

Suppose we are given a unitary oracle U_f for a boolean formula $f\colon \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2$ (section 2.2.2), and we wish to identify some n-bit strings x satisfying $f(x)=1$. U_f acts as $U_f|x\rangle\otimes|y\rangle \rightarrow |x\rangle\otimes|y\oplus f(x)\rangle$. We can convert such an oracle to another U_w as follows :

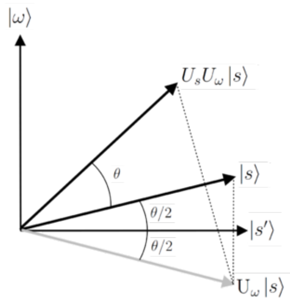
$$|x\rangle\otimes|0\rangle \longrightarrow [I\otimes X] \longrightarrow |x\rangle\otimes|1\rangle \longrightarrow [I\otimes H] \longrightarrow |x\rangle\otimes\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \longrightarrow [U_f] \longrightarrow$$

$$\longrightarrow |x\rangle\otimes\left(\frac{|f(x)\rangle-|1\oplus f(x)\rangle}{\sqrt{2}}\right) = (-1)^{f(x)}|x\rangle\otimes\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \longrightarrow [I\otimes(X^{-1}H^{-1})] \longrightarrow (-1)^{f(x)}|x\rangle\otimes|0\rangle.$$

So $U_w\otimes I \equiv (I\otimes(X^{-1}H^{-1}))U_f(I\otimes(HX))$ acts as an oracle that "recognizes" strings x with $f(x)=1$ by negating the phase : $(U_w\otimes I)|x\rangle|0\rangle \rightarrow (-1)^{f(x)}|x\rangle|0\rangle$. Assuming for simplicity that $f(x)$ is 1 for a single string w. Grover's algorithm starts from a uniform superposition $|s\rangle = 2^{-n/2}\sum_{x\in\mathbb{Z}_2^n}|x\rangle$ and starts increasing the amplitude of $|w\rangle$ while reducing that of others.

Rewriting $|s\rangle = 2^{-n/2}|w\rangle + \sqrt{\frac{2^n-1}{2^n}}|s'\rangle$ where the rest of the terms $|s'\rangle = \sqrt{\frac{1}{2^n-1}}\sum_{x:f(x)\neq 1}|x\rangle$.

From its definition, $U_w = \sum_{x:f(x)\neq 1}|x\rangle\langle x| - |w\rangle\langle w| = I - 2|w\rangle\langle w|$. Geometrically, U_w is a reflection about the hyperplane with $|w\rangle$ as its normal direction, which contains the $|s'\rangle$ direction. The algorithm proceeds by making such successive reflections about $|s\rangle$ and $|s'\rangle$, while never leaving the $|s'\rangle, |w\rangle$ plane. The operator U_s depicted below is the Grover diffusion operator $=2|s\rangle\langle s| - I$. $\cos(\theta/2) = \langle s'|s\rangle = \sqrt{\frac{2^n-1}{2^n}}$. Each pair of reflections cause the state to rotate towards $|w\rangle$ by an angle θ, which brings the optimal number of iterations to about $\Theta(\sqrt{2^n})$. This gives us a quadratic speedup over exhaustive search. Measuring the resulting state in computational basis produces a feasible string with high probability.



Grover's algorithm – geometric view

3.2 Phase estimation

Phase estimation is a technique to estimate the eigenvalue $e^{2\pi i\varphi}$ of a unitary operator U *given its eigenstate* $|x\rangle$ and store the result as $\varphi \in [0, 1)$ accurate up to n qubits.

An effecient implementation of phase estimation exists, provided we can construct oracles for U^{2^j} (2^j applications of U) effeciently, and can condition those oracles on external qubits. It also employs the quantum Fourier transform. We won't go into the details, and will only look at the procedure as a black box.

Given U and an eigenvector $|x\rangle$, i.e $|x\rangle$ can be prepared effeciently, or is available to us, where $U|x\rangle = e^{2\pi i\varphi}|x\rangle$, QPE is itself a quantum circuit P that takes n additional qubits, and acts on $|0\rangle_n \otimes |x\rangle$ to give $|\tilde{\varphi}\rangle_n \otimes |x\rangle$; $\tilde{\varphi}$ being an approximation $2^n\varphi$ to n-bits.

Chapter 5 of [29] is a good resource on the quantum Fourier transform and quantum phase estimation. QPE applied on specific unitaries U directly result in the famous polynomial time algorithms for factoring and the discrete logarithm problem.

3.3 Hamiltonian simulation

The earliest motivations of building a quantum computer were the realization that a such a device could effeciently simulate quantum mechanics much faster than a classical computer. Chapter 4.7 of [29] is an excellent introduction to the topic. Hamiltonian simulation is a fundamental part of simulating physical systems.

The general idea is to simulate Schrödinger evolution by approximating the operator $e^{-i\hat{H}t}$ via quantum gates. This is hard in general, since $\hat{H}$ could be exponentially large.

Usually, simulation is studied for specific classes of Hamiltonians. E.g when $\hat{H}$ is a sum of less complex, more locally acting Hamiltonians $\sum_k \hat{H}_k$. The heart of simulation algorithms for such sums is the Trotter product formula :

$$\lim_{n\to\infty} (e^{iA\Delta t/n} e^{iB\Delta t/n}) = e^{i(A+B)\Delta t}$$

which is verified by expanding the Taylor series. This is valid even when A and B don't commute. Note that in general for two matrices A and B unless $AB = BA$, e^{A+B} need not $=e^A e^B$. We can also derive error bounds like so : $e^{i(A+B)\Delta t} = e^{iA\Delta t} e^{iB\Delta t} + O(\Delta t^2)$ which give us approximation errors made by our simulation.

If the unitary transforms $e^{-i\hat{H}_k\Delta t}$ can be effeciently implemented as gates, we get an approximation for $e^{-i\hat{H}\Delta t}$. Hamiltonians of the kind $H = H_1 \otimes H_2 \ldots \otimes H_k$, also sometimes have effecient gate implementations.

[9] gives an algorithm to effeciently simulate sparse Hamiltonians – ones whose matrix have at most s non-zero elements per row/column. Usually algorithms taking sparse matrices as input assume an oracle that takes (r, i) indices and returns i^{th} non-zero element of the row r. [14] discusses discrete and continuous time quantum random walks on graphs, and the correspondance of the two cases leads to an algorithm for effeciently simulating certain Hamiltonians. Quantum random walks are interesting in their own right, and are further discussed in appendix B.

4 Adiabatic quantum computation

4.1 The quantum adiabatic algorithm

The term *adiabatic* itself has roots in thermodynamics; it refers to processes that don't transfer heat. Classically, such reversible processes are also called *isentropic* – entropy preserving.

The quantum adiabatic algorithm was first presented by Farhi, Goldstone, Gutmann and Sipser in [19]. QAA works with *time-dependent* Hamiltonians $\hat{H}(t)$. The analysis of time resource requirements for the algorithm relies on the *quantum adiabatic theorem*. In it's original form it states – *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 given eigenvalue and the rest of the energy spectrum.*

The system evolves as given by the Schrödinger equation $i\hbar\frac{d}{dt}|\psi(t)\rangle = \hat{H}(t)|\psi(t)\rangle$. For some smooth family of Hamiltonians $\{\hat{H}(t): 0 \leqslant t \leqslant T\}$, denote the instantaneous eigenstates/eigenvalues as $\hat{H}(t)|l;t\rangle = E_l(t)|l;t\rangle$, where $E_l(t)$ is the l^{th} energy level at time $t: E_0(t) \leqslant E_1(t) \leqslant \cdots \leqslant E_{N-1}(t)$. According to the adiabatic theorem, a system in the ground state at time 0 $|\psi(0)\rangle = |0;0\rangle$ stays in the ground state provided T is suffeciently large (i.e the process is suffeciently slow.) More concretely, $\lim_{T\to\infty} \langle l=0, t=T|\psi(T)\rangle = 1$.

As noted in [19], for $g_{\min} \equiv \min_{0\leqslant t\leqslant T}\{E_1(t) - E_0(t)\}$ (the smallest spectral gap), $T > \Theta(1/g_{\min}^2)$ suffices. Intuitively, the smaller the spectral gap, the more likely it is to accidentally transition from the ground state. The trick then is to construct Hamiltonians H_B and H_P on n-qubit systems that have desired ground eigenstates. H_B has a ground state that is easily found and prepared, and the ground state of H_P encodes the solution to a problem of interest, and $\hat{H}(t) = H_B(1 - t/T) + tH_P/T$.

[19] proceeds to show how we can encode binary constraint satisfaction problems as Hamiltonians, and how the above Schrödinger evolution gives a final state $|\psi(T)\rangle$ maximizing the number of satisfied clauses. The problem with using adiabatic computation to solve NP problems is that one usually finds the spectral gap $g_{\min}$ to be exponentially small in problem size N, making $T \in \Omega(\exp(\alpha N^\beta))$. So, while it's unlikely that QAA might solve NP problems exponentially faster than classical algorithms, the constants α and β might be smaller than classically possible.

4.2 The Ising model, and quadratic binary optimization

A particular kind of physical system whose Hamiltonian is of interest, is the one described by the Ising model. It was originally conceived to study magnetism arising out of interacting spins in nearby atoms in a metallic lattice. The configuration of such systems is given by a graph $G(V, E)$ of spin sites V and adjacent interacting pairs of sites E. The (classical) Hamiltonian for this graphical model takes the form $H(\sigma) = -\sum_{(i,j)\in E} J_{ij}\sigma_i\sigma_j - \mu\sum_{i\in V} h_i\sigma_i$, $\sigma \in \{+1, -1\}^{|V|}$. Clearly, for $\mu = 0$, the cut $\{i: \chi_i = +1\}$ for $\chi \in \operatorname{argmin}_\sigma H(\sigma)$ is a solution to a weighted max-cut instance. A quantum version of such a Hamiltonian, encoding this and other NP-hard problems, is given in [28].

The class of quadratic unconstrained binary optimizations (QUBO) that seek to find $\operatorname{argmin}_{x\in\mathbb{Z}_2^n}\{x^TQx\}$, for symmetric $n \times n$ matrices $Q \in \mathbb{S}_n$ also falls under this paradigm, as they are computationally equivalent to the Ising model. Solving this class of problems by quantum annealing is one of the main intended uses of some adiabatic quantum computers. (One such computer is supposedly right here at USC.) Although, quantum annealing might not be exactly the same as implementing the adiabatic algorithm as mentioned above, the two are closely related. It's important to note is that this model of computation is equivalent [3] to the standard gate model discussed earlier.

4.3 A variational approximation algorithm

While QAA aims to solve problems exactly, there do exist families of quantum circuits that can approximate a solution. Farhi, Goldstone and Gutmann [18] present an approximation scheme (QAOA) that takes a parameter p, and constructs a circuit with depth scaling linearly with p, that gives increasingly good approximations for max-SAT instances. QAOA is an example of a *variational algorithm*. It applies a specific unitary transform $U(\beta, \gamma)$ parametrized by $\beta, \gamma \in \mathbb{R}^p$ to generate a state $U(\beta, \gamma)|\psi_0\rangle \to |\psi(\beta, \gamma)\rangle$. The goal is to find optimal parameters $(\beta^\star, \gamma^\star)$ using classical optimizers such that $|\psi(\beta^\star, \gamma^\star)\rangle$ encodes the approximate optimal solution.

$U(\beta,\gamma)$ has a specific form : $U(\beta,\gamma)=\prod_{j\in[p]} e^{-i\beta_j H_B} e^{-i\gamma_j H_P}$. The Hamiltonians H_B and H_P have the same ground states as they do in QAA, and are also called the *mixing* and *problem* Hamiltonians respectively. The convergence guarantee when $p\to\infty$ follows from the fact that larger circuits more closely approximate the adiabatic evolution.

The algorithm proceeds by initializing the parameters (β,γ) and repeatedly updating them. This update is done by preparing $|\psi(\beta,\gamma)\rangle$ using the above unitaries (which are assumed to have effecient gate implementations) and calculating the expectation $\langle\psi(\beta,\gamma)|H_P|\psi(\beta,\gamma)\rangle$. Then a classical optimizer is used to find a new set of parameters (γ',β') for the next step. The choice of optimizer varies between exact applications. Note that the optimizer doesn't have access to β,γ gradients, and must use some gradient-free methods.

This paradigm of hybrid quantum-classical computation using variational algorithms has gained a lot of traction recently, especially since the advent of the so called variational quantum eigensolver (VQE) [30] for estimating the smallest (ground-state) eigenvalue, which has applications in chemistry and physics. One of the advantages of such methods is their applicability on already available, noisy, intermediate scale quantum computers that have limited coherence times and fault tolerance.

5 The HHL algorithm for solving linear systems

Harrow, Hassidim and Lloyd [24] exhibit a quantum algorithm, denoted as the HHL algorithm, to solve $N\leqslant 2^n$ dimensional linear systems of the form $Ax=b$. We will briefly discuss the overview of the algorithm and then go over some its applications and limitations. One of the broad ideas in quantum algorithms involving matrices is to consider them as Hamiltonians, and build circuits resembling them. HHL is an example of this.

5.1 The algorithm

The HHL algorithm encodes vectors b and x by normalizing and mapping them to quantum states $|b\rangle$ and $|x\rangle$. This is done by creating the state $|b\rangle=\sum_{i\in\mathbb{Z}_2^n} b_i|i\rangle$, i.e the components are stored as amplitudes. If A is not Hermitian, the algorithm works on solving $\begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix} y=\begin{pmatrix} b \\ 0 \end{pmatrix}$ to obtain $y=\begin{pmatrix} 0 \\ x \end{pmatrix}$, hence A is assumed Hermitian w.l.o.g. A then has a decomposition $A=\sum_j \lambda_j|u_j\rangle\langle u_j|$ with λ_j as eigenvalues for eigenvectors $|u_j\rangle$. Rewriting $|b\rangle$ in that eigenbasis as $|b\rangle=\sum_j \beta_j|u_j\rangle$, we have $|x\rangle=A^{-1}|b\rangle=\sum_j \lambda_j^{-1}\beta_j|u_j\rangle$, which is the state HHL computes.

To do so, HHL works with the unitary $U=e^{iAt}=\sum_j e^{i\lambda_j t}|u_j\rangle\langle u_j|$, and uses techniques of Hamiltonian simulation from [9], [14] to implement e^{iAt}. Using quantum phase estimation (section 3.2) with U, it prepares the state $\sum_j \beta_j|\tilde{\lambda_j}\rangle|u_j\rangle$. (Approximately. Since the eigenvalues are not always exact, an in-depth analysis takes care of inaccurate phase estimation and its effect on the final state, which is expressed as a double sum in the paper.) Here $\tilde{\lambda_j}$ is an $n-$bit binary approximation of $\lambda_j/2\pi$. If λ_js are large, we can always scale the entire system.

Next, on an auxiliary qubit, a rotation conditional on $|\tilde{\lambda_j}\rangle$ is applied, which results in the state $\sum_j \beta_j|\tilde{\lambda_j}\rangle|u_j\rangle|0\rangle \longrightarrow \sum_j \beta_j|\tilde{\lambda_j}\rangle|u_j\rangle\left\{\sqrt{1-\frac{C^2}{\tilde{\lambda_j}^2}}|0\rangle+\frac{C}{\tilde{\lambda_j}}|1\rangle\right\}$. Then the auxiliary qubit is measured until it's observed to collapse to $|1\rangle$, causing the resulting state to be $\frac{1}{Z}\sum_j C\beta_j\tilde{\lambda_j}^{-1}|\tilde{\lambda_j}\rangle|u_j\rangle|1\rangle$, where the normalization factor $Z=\sqrt{\sum_k |\beta_k^2||C^2|/|\tilde{\lambda}_k^2|}$ is estimated by the probability of observing $|1\rangle$. This is the desired state up to a normalization C/Z.

The running time of the algorithm is lower bounded by the success probability of observing $|1\rangle$, which in turn depends on $|C|\in O(1/\kappa)$. And the phase estimation error translates to a final error ϵ. A rudimentary analysis shows that the running time is $\tilde{O}(\log(N)s^2\kappa^3/\epsilon)$ for s-sparse matrices. An extra factor of κ can be saved by amplitude amplification (section 3.1).

5.2 Applications and limitations

The conception of the HHL algorithm sparked a “mini-revolution” in the field of quantum machine learning. Suddenly there was renewed interest in solving practical problems like least squares fitting, classification and clustering using this exponentially faster algorithm. Similar algorithms with related ideas started popping up. This included solving linear differential equations [8], and various tasks in machine learning [27] and data fitting [33].

As aptly put in [1], HHL is not *exactly* an algorithm for solving linear equations $Ax=b$ in logarithmic time. Rather, it's an algorithm for approximately preparing a quantum superposition $|x\rangle$. The paper also goes on to discuss its limitations, in that HHL guarantees an exponential speedup only when (1) the state $|b\rangle$ can be loaded quickly in the quantum computer's memory, (2) the unitary e^{-iAt} is effeciently implemented as a quantum circuit, and A is sparse (3) A is *well-conditioned*, i.e its condition number $\kappa=|\lambda_{\max}|/|\lambda_{\min}|$ is small; note that classical algorithms like the conjugate gradient method also prefer well-conditioned matrices. (As $\lambda_{\min}\to 0$, the balls are mapped to ellipsoids that become more and more oblong, until they are flat in some direction.) (4) Simply writing x requires linear time, however HHL produces $|x\rangle$ in logarithmically many qubits. The algorithm is useful if we can utilize $|x\rangle$ and may not be when we definitely need x. [1] further shows that despite these limitations, there exist potential applications of HHL.

[4] improves on HHL by giving an algorithm that generalizes amplitude amplification, and uses it to give an improved algorithm for solving linear systems in $\tilde{O}(\kappa\log^3\kappa\log(N)\,\mathrm{poly}(1/\epsilon))$.

[16] improves on HHL by giving an algorithm with runtime $O(\log(1/\epsilon))$. However, as noted in the paper, if the output $|x\rangle$ is needed to sample expectations of the kind $\langle x|M|x\rangle$, the sampling error alone rules out a $\mathrm{poly}(\log(1/\epsilon))$ time algorithm. Still, this is useful when used as a subroutine being called polynomially many times.

6 Semidefinite programming

A semidefinite program (SDP) is a special case of a convex program, with numerous applications in a variety of topics ranging from operations research to combinatorial optimization. SDPs also generalize linear programs. They are an indespensible tool when it comes to the development of approximation algorithms for NP-hard optimization problems, the most famous of which has to be the SDP relaxation of the max cut problem [22] by Goemans and Williamson.

The popular classical algorithms for solving SDPs are mostly based on numerical primal-dual methods like the interior point method [10], which work by applying Newton-Raphson iterations on a pair of (primal, dual) vectors to minimize a barrier function. These methods are bottlenecked by the need to store and invert a large block matrix of Hessians and Jacobians. There also exist other first order methods.

For our purposes, however, we'll start with a seemingly unrelated framework of the *multiplicative weights* (MW) algorithm. This meta-algorithm has been rediscovered multiple times in different fields ranging from machine learning to game theory to constrained optimization. Arora, Kale and Hazan [5] give a great survey on this framework, and our discussion will be based on it. This will eventually lead us to a primal-dual method for solving SDPs [6]. Lastly, we'll gently scratch the surface of a quantum algorithm presented in [12] which is based on these ideas.

6.1 The multiplicative weights algorithm

As introduced in [5], MW is a generalization of the *weighted majority* algorithm which solves the *prediction from expert advice* problem. In this problem, we have n experts, each giving a binary bit of advice; will it or won't it rain today, for example; on every time step for predicting a global observation. The weighted majority algorithm starts by giving an equal weight to every expert. Everyday, it makes decisions based on the weighted sum of the experts' advice, and then penalizes all of the experts that gave the wrong advice that day, by halving their weight. We can prove that the number of mistakes it makes is bounded above by twice the number of mistakes made by the best expert, plus additive factors.

In the general randomized setting, there are a set of n choices, for each of the T rounds, and we are required to make a single choice on every round. After we make that choice, the costs associated with making the decision at time t are revealed as a vector $m^{(t)}$. Our objective is to compute a distribution $p^{(t)}$ over the set of decisions at every time step, so that the expectation $\sum_{t\in[T]}\sum_{i\in[n]} p_i^{(t)} m_i^{(t)} = \sum_{t\in[T]} (p^{(t)}, m^{(t)})$ is close to the cost of a single best decision $\min_i \sum_{t\in T} m_i^{(t)}$ we could have made in hindsight. The algorithm quite simply maintains a weight function $w^{(t)} \in \mathbb{R}^n$ of the decisions at every time step, and declares $p^{(t)} \equiv w^{(t)}/\Phi^{(t)}$, where $\Phi^{(t)} \equiv \sum_{i\in[n]} w_i^{(t)}$. After every decision, the weights are *multiplicatively updated* to suppress costly choices.

Algorithm 1

```
Initialize : Fix $0 < \eta \leqslant 1/2$, and set $w^{(1)} = \mathbf{1}$
$\longrightarrow \forall t \in [T]$: $p^{(t)} \longleftarrow w^{(t)}/\Phi^{(t)}$. Use this distribution to sample a choice
$\longrightarrow$ observe $m^{(t)}$, and penalize costly decisions: $w_i^{(t+1)} \longleftarrow w_i^{(t)}(1 - \eta m_i^{(t)})$
```

Assuming all costs $|m_i^{(t)}| \leqslant 1$, the MW algorithm guarantees that after T rounds, for any decision $i \in [n]$, the following bound holds :

$$\begin{aligned}&\textstyle\sum_{t\in[T]} (m^{(t)}, p^{(t)}) \leqslant \sum_{t\in T} m_i^{(t)} + \eta \sum_{t\in T} |m_i^{(t)}| + \frac{\ln(n)}{\eta}\\ \Longrightarrow &\textstyle\sum_{t\in[T]} (m^{(t)}, p^{(t)}) \leqslant \sum_{t\in T} (m^{(t)} + \eta |m^{(t)}|, p) + \frac{\ln(n)}{\eta}\end{aligned}$$

Note that the factor of two from the deterministic setting is now dropped to one. As noted in [5], the Hedge algorithm [21] of Freund and Schapire is a variation of MW, except the weights are updated with an exponential multiplicative factor :

$$w_i^{(t+1)} \longleftarrow w_i^{(t)} e^{-\eta m_i^{(t)}}$$

to obtain a similar looking regret bound (using $e^{-\eta x} \leqslant 1 - \eta x + \eta^2 x^2 \quad \forall |\eta x| \leqslant 1$):

$$\forall i \in [n]: \textstyle\sum_{t\in[T]} (m^{(t)}, p^{(t)}) \leqslant \sum_{t\in T} m_i^{(t)} + \eta \sum_{t\in T} \{\sum_{j\in[n]} (m_j^{(t)})^2 p_j^{(t)}\} + \frac{\ln(n)}{\eta}$$

The authors of [21] go on to apply this to their *Gö*del award winning conception of AdaBoost.

6.2 Multiplicative weights as constructive LP duals

[5] show the connection between MW and LPs by giving the example of learning a separating hyperplane classifier. This is a classic task in machine learning. As they note, the following can be regarded as a "constructive" version of LP duality. When solving constrained optimization problems, each decision represents a constraint, with costs representing the corresponding potential function. Iteratively re-weighting these costs can now be seen as a game-theoretic version of Lagrangian optimization.

Suppose we are given m points of labelled data (x_i, y_i), $x_i \in \mathbb{R}^n$, $y_i \in \{\pm 1\}$, and our objective is to find a separating hyperplane w satisfying $\operatorname{sign}((w, x_i)) = y_i$, or equivalently $(y_i x_i, w) \geqslant 0$ for all x_i. Assuming w.l.o.g, $w \in \Delta^n$, where Δ^n is the probability simplex $\{\alpha \in \mathbb{R}^n: \sum_i \alpha_i = 1, \alpha \succcurlyeq 0\}$, and renaming $y_i x_i = z_i$, the problem reduces to an LP feasibility problem $\exists ? w \in \Delta^n: \forall j \in [m] (z_j, w) \geqslant 0$.

Assuming that there is a large margin solution; i.e $\exists \varepsilon > 0, w^\star: \forall j \in [m]: (z_j, w^\star) \geqslant \varepsilon$, MW can be used to solve this as follows. Define $\rho \equiv \max_j \|z_j\|_\infty$, and set $\eta \longleftarrow \varepsilon/2\rho$, and set the costs as $m_i^{(j)} = (z_j)_i/\rho$. $(|m_i^{(j)}| \leqslant 1)$

In each round t, we let w to be the distribution $p^{(t)}$ generated by MW, and look at an unsatisfied constraint (a misclassified example) j, and use $m^{(j)}$ for weight update. We keep doing this until we find a feasible point. As shown in [5], we can safely terminate after $\lceil 4\rho^2 \log(n)/\varepsilon^2 \rceil$ iterations.

6.3 The matrix multiplicative weights algorithm

The MW algorithm can be extended to get a version that solves SDP feasibility (and hence by convexity, optimization) using a similar exponential weight update rule.

The decisions now correspond to unit vectors $v^{(t)} \in \mathbb{R}^n$, ${v^{(t)}}^T v^{(t)} = 1$. The costs are given as a matrix $M^{(t)} \in \mathbb{R}^{n \times n}$, and the cost of each decision is ${v^{(t)}}^T M^{(t)} v^{(t)}$. Just like before, we assume all singular values of $M^{(t)}$ are $\leqslant 1$. The objective at each time step is to compute a distribution $\mathcal{D}^{(t)}$ over our choices, such that the expected cost $\sum_t \mathbb{E}_{v \sim \mathcal{D}^{(t)}}[v^{(t)T} M^{(t)} v^{(t)}]$ is close to the best fixed decision $v^T(\sum_{t \in T} M^{(t)}) v$, which is just the smallest eigenvalue of $\sum_{t \in T} M^{(t)}$.

Denote the inner product on symmetric matrices $C, X \in \mathbb{S}_n$ as $\langle C, X \rangle = \mathrm{Tr}[CX] = \sum_{ij} C_{ij} X_{ij}$, where Tr[.] is the trace operator. $\mathbb{E}_{v \sim \mathcal{D}^{(t)}}[v^{(t)T} M^{(t)} v^{(t)}] = \mathbb{E}_{v \sim \mathcal{D}^{(t)}}[\mathrm{Tr}[v^{(t)T} M^{(t)} v^{(t)}]] = \mathbb{E}_{v \sim \mathcal{D}^{(t)}}[\mathrm{Tr}[M^{(t)} v^{(t)} v^{(t)T}]] = \mathrm{Tr}[M^{(t)} \mathbb{E}_{v \sim \mathcal{D}^{(t)}}[v^{(t)} v^{(t)T}]] = \langle M^{(t)}, \mathbb{E}_{v \sim \mathcal{D}^{(t)}}[v^{(t)} v^{(t)T}] \rangle$.

Denote $\mathbb{E}_{v \sim \mathcal{D}^{(t)}}[v^{(t)} v^{(t)T}] = P^{(t)}$. It's easy to see that $P^{(t)} \succcurlyeq 0, \mathrm{Tr}[P^{(t)}] = 1$. $P^{(t)}$ is a hence a density matrix (appendix A). And it's exactly what MMW computes at each step. Any distribution $\mathcal{D}^{(t)}$ with this density suffices. The eigendecomposition of $P^{(t)}$ gives one such discrete distribution.

Algorithm 2

```
Initialize : Fix $0 < \eta \leqslant 1/2$, and set $W^{(1)} = I$
$\longrightarrow \forall t \in [T]$: $P^{(t)} \longleftarrow W^{(t)} / \Phi^{(t)}$, with $\Phi^{(t)} \equiv \mathrm{Tr}[W^{(t)}]$.
$\longrightarrow$ observe $M^{(t)}$, and update weights as : $W_i^{(t+1)} \longleftarrow W_i^{(t)} e^{-\eta M^{(t)}}$
```

Using an additional inequality $\mathrm{Tr}[e^{A+B}] \leqslant \mathrm{Tr}[e^A e^B]$, we have a matrix version of the familiar regret bound: $\forall v \in \mathbb{R}^n, v^T v = 1$

$$\sum_{t \in [T]} \langle M^{(t)}, P^{(t)} \rangle \leqslant \sum_{t \in [T]} v^T M^{(t)} v + \eta^2 \sum_{t \in [T]} \langle (M^{(t)})^2, P^{(t)} \rangle + \frac{\ln(n)}{\eta}$$

We'll consider the following form of feasibility:$\exists ? X \in \Delta_n^n : \forall j \in [m] \; \langle A_j, X \rangle \geqslant 0$, where Δ_n^n is the set of density matrices $\{X \in \mathbb{S}_n : \mathrm{Tr}[X] = 1, X \succcurlyeq 0\}$. (This shape is also called a *spectrahedron*.) When all matrices are diagonal, this is the same is LP feasibility discussed earlier.

Assuming there is a large margin solution: $\exists \varepsilon > 0 \, s.t \; \langle A_j, X \rangle \geqslant \varepsilon$, and defining $\rho \leftarrow \max_j \|A_j\|$ (largest operator norm), setting $\eta \longleftarrow \varepsilon / 2\rho$ and the costs as $M^{(t)} = A_j / \rho$ for some unsatisfied constraint j, we run the same procedure, to get an exactly analogous algorithm. The running time is again bounded by $\lceil 4\rho^2 \log(n) / \varepsilon^2 \rceil$.

6.4 A fast approximate primal-dual algorithm for SDPs

We'll now briefly discuss the faster version of this algorithm presented in [6]. The primal SDP is :

$$\max \langle C, X \rangle \; s.t \forall j \in [m] : \langle A_j, X \rangle \leqslant b_j, X \succcurlyeq 0$$

and its dual is

$$\min \langle b, y \rangle \; s.t \sum_{j \in [m]} y_j A_j \succcurlyeq C, y \succcurlyeq 0$$

Optimization is reduced to feasibility using binary search. Assume $A_1 = I, b_1 = R \Longrightarrow \mathrm{Tr}[X] \leqslant R$, which serves as a bound for binary search. Every feasibility subproblem is to construct either a primal feasible PSD matrix X with value $> \alpha$, or a dual feasible vector y with value $\leqslant \alpha(1+\delta)$ for arbitrarily small $\delta > 0$. The algorithm mimics MMW by generating iterates $X^{(t)}$ over $1 \leqslant t \leqslant T$ time steps. At the center of this algorithm is an oracle $\mathcal{O}$ which is used to generate these iterates. $\mathcal{O}$ tries to find a dual vector $y \in D_\alpha = \{y : y \succcurlyeq 0 : \langle b, y \rangle \leqslant \alpha\}$ subject to the constraints $\sum_{j \in [m]} \langle A_j, X^{(t)} \rangle y_j - \langle C, X^{(t)} \rangle \geqslant 0$. Note that this is an LP with a single non-trivial constraint.

If there is no such $y \in D_\alpha$, we have a feasible X with value $\geqslant \alpha$. And if $\mathcal{O}$ succeeds in finding such a y, $X^{(t)}$ is either primal infeasible or has value $\leqslant \alpha$. The algorithm's progress is measured using the *width property* ρ of the oracle $\mathcal{O}$: $\rho \equiv \sup_{y \in D_\alpha} \|\sum_j y_j A_j - C\|$ ($\|.\|$ denotes operator norm)

The authors of [6] prove that if algorithm 3 proceeds for $T \geqslant \frac{8\rho^2 R^2 \ln(n)}{\delta^2 \alpha^2}$ iterations, then $y^\star \equiv \frac{\delta\alpha}{R} \boldsymbol{e}_1 + \frac{1}{T} \sum_{t \in [T]} y^{(t)}$ is a dual feasible solution with objective at most $(1+\delta)\alpha$. Thus, we get a feasibility subroutine that we can use inside the binary search. Note that R must be known *a priori*. Moreover they obtain fast approximation algorithms for several optimization problems, including one for the SDP relaxation of max-cut.

Algorithm 3

```
Initialize: $W^{(1)} \leftarrow I$, $\varepsilon \longleftarrow \delta\alpha/2\rho R$, $\varepsilon' \longleftarrow -\ln(1-\varepsilon)$
$\forall t \in [T]$:
Define $X^{(t)} \equiv RW^{(t)}/\mathrm{Tr}[W^{(t)}]$
Invoke $\mathcal{O}(X^{(t)})$ to get $y^{(t)}$. If it fails, stop and output $X^{(t)}$ as a feasible solution with value $>\alpha$
else, set the MMW "costs" as $M^{(t)} \longleftarrow \big(\sum_{j\in[m]} A_j y_j^{(t)} - C + \rho I\big)/2\rho$
and update the weights as $W^{(t+1)} \longleftarrow W^{(t)} e^{-\varepsilon' M^{(t)}}$
```

6.5 A quantum algorithm for solving SDPs

[12] gives a quantum algorithm based on the matrix multiplicative weights method, with a claimed worst-case running time of $\Theta(\sqrt{nm}s^2\mathrm{poly}(\log(n), \log(n), R, r, 1/\delta))$. As before, parameter R is an upper bound on $\mathrm{Tr}[X]$ for primal feasible X. The authors also reduce the dual SDP to the case where $b \succcurlyeq \mathbf{1}$, r is an upper bound on $\mathbf{1}^T b$. δ is an additive error tolerance for the dual solution as before. Note however that the running time analyses show the algorithm to be prohibitively costly in terms of R and δ. The authors also believe that this can be mitigated significantly.

Of course, simply writing the optimal primal/dual takes $\Omega(\min(n^2, m))$ time, so the problem is framed instead as the following : given $A_1, A_2 \ldots A_m, A_{m+1} \equiv C$ approximate the optimal value of the above primal and/or dual SDPs. The quantum algorithm is also required to produce an estimate of $\|y\|_1$ and/or $\mathrm{Tr}[X]$, and also be able to generate samples from the *distribution* $p \equiv y/\|y\|_1$ and/or from the density $\rho \equiv X/\mathrm{Tr}[X]$. Like in HHL, the matrices are taken to be s-sparse, and are given by an oracle that takes indices $j \in [m+1], k \in [n], l \in [s]$ and compute the l^{th} non-zero element $\mathrm{nz}_{jk}(l)$ of the k^{th} row of A_j as : $|j, k, l, z\rangle \longrightarrow |j, k, l, z \oplus \mathrm{nz}_{jk}(l)\rangle$.

One of the main ideas in [12] is to use "Gibbs samplers" – quantum circuits that given such an oracle O_H for the entries of a an s-sparse Hamiltonian H produce a state $|\psi\rangle$ with density resembling a Gibbs state $|\psi\rangle\langle\psi| \approx e^H/\mathrm{Tr}[e^H]$. Gibbs sampling is a widely studied topic, the most prominent example of which is the work on a quantum version of the Metropolis algorithm [31], [35].

The authors noticed that applying amplitude amplification (section 3.1) to Gibbs samplers straightaway gives a quadratic speedup w.r.t n. In fact, if the Gibbs sampler has some really nice properties, we can get even exponential speedups. Furthermore, they claim that replacing the oracle $\mathcal{O}$ of [6] with such samplers is also possible, and gives a quadratic speedup w.r.t m.

The main contribution can thus be summarized as follows: using these samplers allows us to quickly compute the multiplicative weight updates, hence the quantum speedup. Several improvements have since been proposed [11], [32].

7 Summary

We explored some quantum algorithms encountered in combinatorial and constrained optimization. The reader must have noticed the importance of Hamiltonians (section 3.3) in constructing quantum algorithms that handle matrices. Which is why we have some additional discussions in appendix B on a connection between Hamiltonians and quantum random walks. Hamiltonians with sparse entries in particular are the ones that are amenable to quantum speedups.

Something worth emphasizing is the interdisciplinary nature of the discussed topics. Most of these developments wouldn't have been possible if not for the collaboration between people working in different fields, ranging from theoretical computer science, to physics, to engineering. This has to inspire awe in anyone interested in quantum computing.

The author hopes that this paper serves as a good starting point for exploring optimization algorithms from a quantum point of view, and thanks the reader for their interest.

Bibliography

[1] Scott Aaronson. Read the fine print. *Nature Physics*, 11:291–293, 04 2015.

[2] Dorit Aharonov, Andris Ambainis, Julia Kempe, and Umesh Vazirani. Quantum walks on graphs. 2000.

[3] Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. 2004.

[4] Andris Ambainis. Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations. 2010.

[5] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012.

[6] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. *J. ACM*, 63(2), may 2016.

[7] C. H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17(6):525–532, 1973.

[8] Dominic W Berry. High-order quantum algorithm for solving linear differential equations. *Journal of Physics A: Mathematical and Theoretical*, 47(10):105301, feb 2014.

[9] Dominic W. Berry, Graeme Ahokas, Richard Cleve, and Barry C. Sanders. Efficient quantum algorithms for simulating sparse hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, dec 2006.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, England, mar 2004.

[11] Fernando G. S. L. Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M. Svore, and Xiaodi Wu. Quantum sdp solvers: large speed-ups, optimality, and applications to quantum learning. 2017.

[12] Fernando G. S. L. Brandao and Krysta Svore. Quantum speed-ups for semidefinite programming. 2016.

[13] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. 2002.

[14] Andrew M. Childs. On the relationship between continuous- and discrete-time quantum walk. *Communications in Mathematical Physics*, 294(2):581–603, oct 2009.

[15] Andrew M. Childs, Edward Farhi, and Sam Gutmann. *Quantum Information Processing*, 1(1/2):35–43, 2002.

[16] Andrew M. Childs, Robin Kothari, and Rolando D. Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, jan 2017.

[17] Christopher M. Dawson and Michael A. Nielsen. The solovay-kitaev algorithm. *Quantum Info. Comput.*, 6(1):81–95, jan 2006.

[18] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. 2014.

[19] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. 2000.

[20] Edward Farhi and Sam Gutmann. Quantum computation and decision trees. *Physical Review A*, 58(2):915–928, aug 1998.

[21] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[22] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, nov 1995.

[23] Lov K. Grover. Quantum computers can search rapidly by using almost any transformation. *Physical Review Letters*, 80(19):4329–4332, may 1998.

[24] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15), oct 2009.

[25] J Kempe. Quantum random walks: an introductory overview. *Contemporary Physics*, 44(4):307–327, jul 2003.

[26] R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3):183–191, 1961.

[27] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. 2013.

[28] Andrew Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2, 2014.

[29] Michael A Nielsen and Isaac L Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, England, dec 2010.

[30] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1), jul 2014.

[31] K. Temme, T. J. Osborne, K. G. Vollbrecht, D. Poulin, and F. Verstraete. Quantum metropolis sampling. *Nature*, 471(7336):87–90, mar 2011.

[32] Joran van Apeldoorn, Andrá s Gilyén, Sander Gribling, and Ronald de Wolf. Quantum SDP-Solvers: better upper and lower bounds. *Quantum*, 4:230, feb 2020.

[33] Nathan Wiebe, Daniel Braun, and Seth Lloyd. Quantum algorithm for data fitting. *Physical Review Letters*, 109(5), aug 2012.

[34] Noson S. Yanofsky. An introduction to quantum computing. 2007.

[35] Man-Hong Yung and Alá n Aspuru-Guzik. A quantum–quantum metropolis algorithm. *Proceedings of the National Academy of Sciences*, 109(3):754–759, jan 2012.

Appendix A Density operators

There is an alternate formulation [29] of the postulates which ascribe quantum systems with a *density operator* ρ acting on the corresponding Hilbert space. For *pure* states $|\psi\rangle$, the density operator is given by $\rho \equiv |\psi\rangle\langle\psi|$. The advantage in using the density operator formalism is its ability to describe *ensembles* of pure states. Physical systems with exactly the state $|\psi\rangle$ are difficult to prepare, and we might have to work with a system having state $|\psi_i\rangle$ with probabilities p_i. The ensemble $\{p_i, |\psi_i\rangle\}$ is defined to have density $\rho \equiv \sum_i p_i |\psi_i\rangle\langle\psi_i|$. A state is hence *pure* when only one of the $p_i s$ is 1.

E.g consider a qubit we know to have either state $|0\rangle$ or $|1\rangle$ with 1/2 probability. The density is then $\frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Notice that this isn't the same as saying the qubit is in the pure superposition state $\frac{|0\rangle + |1\rangle}{\sqrt{2}}$, which has density $\frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$. Densities are characterized by positive definite operators with trace 1. Different ensembles can give rise to the same density operator, which is what ultimately governs measurement statistics.

Analogous to the Schrödinger equation $i\hbar\frac{d}{dt}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle$ describing time evolution of state vectors, the von Neumann equation $i\hbar\frac{\partial\rho}{\partial t} = [\hat{H}, \rho]$ describes time evolution of densities. Here $[\hat{H}, \rho]$ denotes the commutator $\hat{H}\rho - \rho\hat{H}$. As expected, for time independent Hamiltonians $\rho(t) = e^{-i\hat{H}t/\hbar}\rho(0)e^{i\hat{H}t/\hbar}$. I.e a unitary transform U maps densities as: $\rho \longrightarrow U\rho U^\dagger$.

Appendix B Quantum random walks

Random walks are powerful tools when it comes to analyzing and designing randomized algorithms. Quantum random walks sometimes have an advantage over classical random walks. A great introductory survey on this topic is [25], and our discussion will be based around it. This section is in the appendix because of the arguably tangential nature of this topic.

Classical random walks are quite invaluable in theoretical computer science. They provide a general paradigm for sampling from exponentially large spaces. Important properties of random walks that affect their usage in algorithms are their mixing times and hitting times [2]. Quantum random walks behave quite differently than their classical counterparts in these regards, which explains their role in obtaining faster quantum algorithms. We'll go over some basics of quantum random walks. We'll also glance over a connection between random walks and Hamiltonians.

B.1 A discrete quantum random walk

As an example, we'll briefly analyse the simplest of discrete random walks. Consider the state of a quantum particle varying over two properties – its "spin" $\{\uparrow,\downarrow\}$, and its position on a 1D lattice : $\mathbb{Z}$. The spin state of an arbitrary superposition is hence a vector in $\mathcal{H}_S=\{\alpha|\uparrow\rangle+\beta|\downarrow\rangle:\alpha,\beta\in\mathbb{C}, |\alpha|^2+|\beta|^2=1\}$. Similarly, the superpositions over positions are vectors in $\mathcal{H}_P=\{\sum_{x\in\mathbb{Z}}\alpha_x|x\rangle: \sum_x|\alpha_x|^2=1\}$. The complete state description of the particle is a vector $|s\rangle\otimes|p\rangle\in\mathcal{H}_S\otimes\mathcal{H}_P$.

Define the "coin-flip operator" $C\equiv H\otimes I$, H being the familiar Hadamard; and "conditional-shift operator" $S=|\uparrow\rangle\langle\uparrow|\otimes\sum_{x\in\mathbb{Z}}|x+1\rangle\langle x|+|\downarrow\rangle\langle\downarrow|\otimes\sum_{x\in\mathbb{Z}}|x-1\rangle\langle x|$. The particle jumps right if its spin is up, and left if down: $S(|\uparrow/\downarrow\rangle\otimes|x\rangle)=(|\uparrow/\downarrow\rangle\otimes|x\pm1\rangle)$.

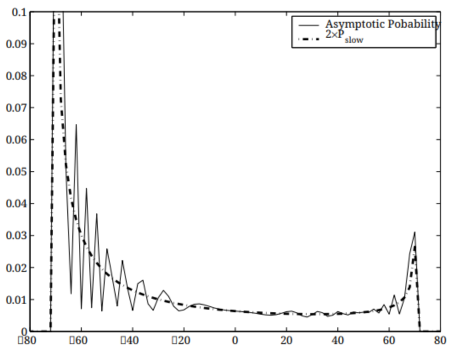


FIG. 5: The probability distribution of the quantum random walk with Hadamard coin starting in $|\downarrow\rangle\otimes|0\rangle$ after $T=100$ steps. Only the probability at the even points is plotted, since the odd points have probability zero. The dotted line gives a long-wavelength approximation (labeled P_{slow} since it keeps only the slowly varying frequencies [15]), clearly showing the bi-modal character of the distribution.

(a) An asymmetric quantum walk

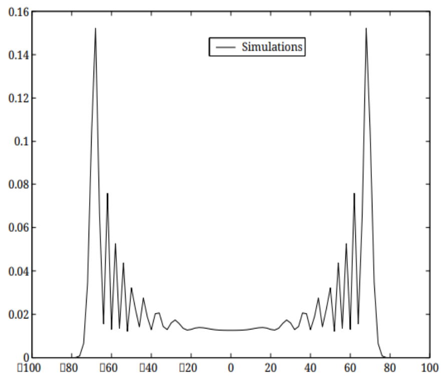


FIG. 6: The probability distribution obtained from a computer simulation of the Hadamard walk with a symmetric initial condition [15]. The number of steps in the walk was taken to be 100. Only the probability at the *even* points is plotted, since the odd points have probability zero.

(b) A symmetric walk.

Figure 2. Simulations of the discrete quantum random walk

For a particle in superposition, S acts linearly, and the resulting amplitudes may add or cancel each other. Consider T alternate applications of S and C applied to an initial state $|\downarrow\rangle$; measuring the position after each time step collapses the state to a definite displacement x. The resulting behaviour is identical to the familiar classical random walk with the resulting statistics resmbling the Gaussian approximation of the binomial distribution. I.e for large T the variance scales as $\sigma^2\sim T$, while the mean is 0.

However, if we don't measure the position at every step, the interferences over all possible paths after T steps produce a probability distribution over $\mathbb{Z}$ that can be skewed, and can even be bimodal . The distribution in figure 2 (a) is a result of $(SC)^T(|\downarrow\rangle\otimes|0\rangle)$. C is asymmetric, and prefers $|\downarrow\rangle$. Figure 2 (b) shows the symmteric results of $(SC)^T\Big(\Big(\frac{|\uparrow\rangle+i|\downarrow\rangle}{\sqrt{2}}\Big)\otimes|0\rangle\Big)$. It can be shown that this quantum walk has a variance that scales as $\sigma^2\sim T^2$.

B.2 Continuous time random walks

Viewing $\mathbb{Z}$ as line graph, we can generalize the above to arbitrary graphs to some extent. [25] cites some studies in this regard, and discusses it in some detail. The procedure of coin flipping and measuring is not really needed for continuous time random processes.

Borrowing the following from Wikipedia, we analyze a single spin-free quantum particle with mass m and a 1D freedom of position x. Its position at time t being completely determined by the wave function $\psi(x,t)$ satisfying the zero-potential Schrödinger equation : $i\hbar\frac{\partial\psi}{\partial t} = \frac{\hat{p}^2\psi}{2m}$, where $\hat{p} = -i\hbar\frac{\partial\psi}{\partial x}$ is the 1D momentum observable. $\Rightarrow i\hbar\frac{\partial\psi}{\partial t} = \frac{-\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2}$.

Discretizing x as $\mathbb{Z}_{\Delta x} \equiv \{\dots, -2\Delta x, -\Delta x, 0, \Delta x, 2\Delta x, \dots\}$, the double derivative is replaced by $\frac{\partial^2\psi}{\partial x^2} \to \frac{\psi((j+1)\Delta x, t) - 2\psi(j\Delta x, t) + \psi((j-1)\Delta x, t)}{\Delta x^2} \equiv \frac{L_{\mathbb{Z}}\psi(j\Delta x, t)}{\Delta x^2}$, giving us the evolution law :
$i\frac{\partial\psi}{\partial t} = -\omega_{\Delta x} L_{\mathbb{Z}_{\Delta x}}\psi$. $\omega_{\Delta x} \equiv \frac{\hbar}{2m\Delta x^2}$ is a constant, and $L_{\mathbb{Z}_{\Delta x}}$ is the *Laplacian* of the line graph $\mathbb{Z}_{\Delta x}$.
For arbitrary graphs $G(V,E)$, the graph Laplacian is defined as $L_G \equiv D_G - A_G$, with D_G the degree matrix (diagonal of degrees) and A_G the adjacency matrix.

In continuous time random walks on graph G, the evolution law is given $\frac{\partial\psi}{\partial t} = -i\omega L_G\psi$, where the ωL_G itself acts as the Hamiltonian [25] with the unitary $U(t) = e^{-i\omega L_G t}$. This was the key idea of Farhi and Gutmann [20], which they use to study decision trees, effectively considering quantum computing as a continuous time random walk. Furthermore, [15] exhibits a finite graph with an exponential separation in *expected hitting times*.

There are a few issues with defining hitting and mixing times in quantum vs. classical walks. Consider the fact that any classical random walk on a finite graph approaches a stationary distribution $p^{(t)}$ regardless of the initial distribution $p^{(0)}$ – it effectively "loses memory". This is not possible for reversible unitary transforms. [25] explains how we can still analyze such dynamics, using the *Cesàro average* distribution instead : $c^{(t)} \equiv \sum_{\tau \leqslant s} p^{(\tau)}$ for some $0 < s \leqslant t$ chosen uniformly at random.