Hardness of Approximation of Quantum Max-Cut

Joseph Carolan, Suchetan Dontha

December 10, 2022

Abstract

In this work, we lay out the fundamentals of quantum computing in the context of complexity theory, then detail current results about hardness of approximation of Quantum Max-Cut. We assume no prior knowledge about quantum computing or the Max-Cut problem, though additional resources may be needed to gain a more complete understanding.

1 Quantum Computing Fundamentals

Quantum computing proposes a model of computation that can potentially provide exponential speedups over classical models of computation (such as regular and probabilistic Turing machines). In particular, this poses a challenge to the strong Church-Turing thesis, which stipulates that any physically reasonable computation device can be simulated by a Turing machine with polynomial overhead. For example, Shor’s algorithm provides a way to factor integers in polynomial time on a quantum computer, a task that, despite great effort, has no efficient classical solution. The purpose of this section is to provide a crash course on the fundamentals of quantum computing as they relate to complexity theory.

1.1 Quantum Computing Model

As it relates to this paper, we can think of a quantum computer as a generalization of a classical randomized computer. To begin, let us define a formalism for talking about randomized circuits. We will consider randomized circuits which act on $n$ bits (including additional storage bits in the initial count), and whose state remain in $n$ bits. A fact which will be important to us is that the ”not” and ”control not” gates (called $X, CX$ respectively) are universal for deterministic classical computation. It follows that, to get universal randomized computation all we will need is a ”randomize” gate, which takes a bit and applies $X$ with 50% probability (called an $R$ gate). Now, we conceptualize the state of the computer as a vector $\vec{v}$ of the form

$$\vec{v} \in \mathbb{R}^{2^n}$$

$$\forall v_i, v_i \geq 0$$

$$\sum_i v_i = 1$$

Our convention will be that a basis vector $\vec{e}_m$, which has a 1 in the $m$-th index corresponds to the bit string given by $m$ in binary. A probabilistic mixture of multiple bit strings is represented by a vector with multiple non-zero entries corresponding to probabilities; this is why the values must
sum to 1. We now note that the gates we wrote down all correspond to linear transformations on such vectors. We can write the matrices (noting that $CX$ acts on 2 bits, the rest on 1)

$$R = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$  \hspace{1cm} (4)$$

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$  \hspace{1cm} (5)$$

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$  \hspace{1cm} (6)$$

We can construct gates acting on the $i$-th bit in an $n$ bit state by tensoring the appropriate number of identities, e.g. flipping the $i$-th bit corresponds to the gate

$$X_i = \left( \bigotimes_{0 \leq j < i} I \right) \otimes X \otimes \left( \bigotimes_{i < j < n} I \right)$$  \hspace{1cm} (7)$$

Now, a circuit $C$ can be thought of as a matrix $M_C$ which is $2^n \times 2^n$; circuits which we can reach as the product of $\text{poly}(n)$ many gate-matrices correspond to efficient circuits. We can think of the actual output of $C$ given starting string $s$, as being drawn from the distribution given $M_C \vec{e}_s$.

We can now interpret quantum computing as a generalization of randomized computing in the following way. We will say that a quantum circuit acts on $n$ qubits (quantum bits) and produces an output state on $n$ qubits. Now, instead of keeping track of the randomized state by a real vector, we will have a complex vector denoted $|\psi\rangle$ (the meaning of $|i\rangle$ is the $i$-th column basis vector). This will be subject to

$$|\psi\rangle \in \mathbb{C}^{2^n}$$

$$= \sum_{i=0}^{2^n-1} \alpha_i |i\rangle \sum_i |\alpha_i|^2 = 1 \hspace{1cm} (8)$$

One can see by standard linear algebra that the set of linear operations preserving these properties for all $|\psi\rangle$ are the unitary matrices of size $\mathbb{C}^{2^n} \times \mathbb{C}^{2^n}$. We will state without proof that the following one or two qubit unitaries can be considered atomic gates

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$  \hspace{1cm} (10)$$

$$P(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\theta} \end{pmatrix}$$  \hspace{1cm} (11)$$

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$  \hspace{1cm} (12)$$

We can "physically" interpret the state as describing a probability distribution on bit string states, with probability of string $i$ given by $|\alpha_i|^2$. However, unlike in probabilistic computing we can now have "destructive interference", wherein multiple computation paths can have some negative and
some positive amplitudes such that over-all they cancel out. Usually, the input and output of a quantum program is a classical string, so we feed in a basis vector and measure at the end, obtaining each state with probability given by the rule above.

A "quantum program" is just an ordered list of these operations, along with the qubits each acts on, and efficiently quantum computable functions are ones which have an efficient quantum algorithm (say that gets the right answer with probability at least $2/3$). Efficient quantum programs are a super-set of efficient classical programs, as they contain in their gate set a $CX$ and $X$ gate (this is non-obvious from our set given; though it is true). Additionally, if we consider applying an $H$ to a pure qubit and then immediately measuring, we obtain a random output. In this way, we can see that efficient quantum programs are also a superset of efficient random programs. It may not be obvious that they are more powerful than random programs, as the only novel ability discussed so far is destructive interference. We will see in later sections how this property can be exploited for computational speed-up. When a vector $|\psi\rangle$ has many non-zero entries, it is referred to as a "coherent superposition", and it is important to understand that this is fundamentally different from a probabilistic mixture. A state like the following

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Is a superposition state. We note that, on measurement, qubit 0 and qubit 1 will both be randomly distributed, but perfectly correlated such that they have the same value. In addition, even if we apply an $H$ gate to each qubit before measuring we see that they are still perfectly correlated; in this sense there is no way to "scramble away" the correlation without measuring. This property is called entanglement, and it is in some sense stronger than classical correlation; we will use it extensively when constructing quantum algorithms.

1.2 Quantum Algorithms

As previously mentioned, quantum computing provides the potential for exponential speedup over currently known classical algorithms. The purpose of this section is to discuss the details of a few quantum algorithms that demonstrate this advantage.

1.2.1 Grover’s Algorithm

Grover’s algorithm provides a quantum speedup for the problem of unstructured search. Suppose we have some function $f : \{1, 2, \ldots N\} \rightarrow \{0, 1\}$ and there exists a unique $a \in \{1, 2, \ldots N\}$ with $f(a) = 1$. Given query access to the function $f$, the unstructured search problem asks us to determine this value $a$. Classically, we can trivially deduce that this task requires $O(N)$ queries to $f$. However, Grover’s algorithm lets us determine $a$ with some non-zero probability in just $O(\sqrt{N})$ queries.

One of the more interesting consequences of this result is that it provides a speedup for solving the NP-Complete problem SAT. The SAT problem asks the following: given an $n$ variable boolean formula $\phi$, does there exist an assignment $a$ so that $\phi(a) = 1$. Classically, it is not known how to solve this problem in time much better than $\text{poly}(n)2^n$ using the trivial brute force algorithm. However, we can immediately see that with Grover’s algorithm that a quantum computer can solve SAT in time $\text{poly}(n)2^{n/2}$ by brute force.

Grover’s algorithm is best described geometrically. Consider an $n$-qubit register, and let

$$u = \frac{1}{2^{n/2}} \sum_{x \in \{0, 1\}^n} |x\rangle$$

be the uniform state vector on $n$ qubits. Then the angle between $u$ and $|a\rangle$ is given by the inverse cosine of their inner product $\langle u, |a\rangle$. Since this positive, we can express this
angles as $\frac{\pi}{2} - \theta$ where $\sin \theta = \frac{1}{\sqrt{2}}$. Hence, assuming $n$ is large, $\theta \approx \frac{1}{2} 2^{n/2}$ since $\sin \theta \approx \theta$ for small $\theta$. The algorithm starts in the state $w = u$ and at each step transforms $w$ closer to $|a\rangle$ by an angle of $2\theta$. Thus in $O(1/\theta) = O(2^n/2)$ steps, we will arrive at a state that has inner product with $|a\rangle$ greater than $1/2$, which upon measuring will yield a with probability at least $1/4$. See Figure 1 for a visual intuition.

Figure 1: A visualization of Grover’s algorithm

1.2.2 Shor’s Algorithm

As previously mentioned, Shor’s algorithm provides an efficient solution for the integer factorization problem. The integer factorization problem asks to find, for a given integer $N$, the set of all prime factors of $N$ (i.e. the prime numbers that divide $N$). Efficient in this context means in particular that we can find a solution in time polynomial of the description of $N$, so $poly(\log(N))$ time. Although the factorization problem has been thought about for hundreds of years, an efficient classical solution for the problem does not exist yet.

The actual quantum aspect of Shor’s algorithm solves the problem of order finding. The order finding problem asks the following: Given a number $A$ with $\text{gcd}(A, N) = 1$, find the order of $A \mod N$, which is defined as the smallest positive integer $r$ such that $A^r \equiv 1 \mod N$. Using basic number theory, the problem of integer factorization can be reduced to order finding.

The order finding algorithm utilizes a widely used protocol called the Quantum Fourier Transform (QFT). Put simply, the QFT maps a state $|x\rangle = \sum_{i=0}^{N-1} \alpha_i |i\rangle$ to the state $\sum_{i=0}^{N-1} \beta_i |i\rangle$ where

$$\beta_i = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \alpha_j \omega_N^{ij} \quad \omega_N = e^{2\pi i / N}.$$  

It is possible to efficiently compute the QFT on a quantum computer using just $O(\log^2(N))$ elementary gates.

Order Finding Algorithm

The order finding algorithm proceeds as follows, which yields the order $r$ with probability at least $1/poly(\log(N))$: 
1. Prepare the initial state
\[
\left( \frac{1}{\sqrt{M}} \sum_{x \in \mathbb{Z}_M} |x\rangle \right) |0\rangle^\otimes n
\]
where \( M \geq 2^{3\log N} \)

2. Perform the transformation \(|x\rangle |y\rangle \rightarrow |x\rangle |y \oplus (A^x \mod N)\rangle\). This yields the state
\[
\frac{1}{\sqrt{M}} \sum_{x \in \mathbb{Z}_M} |x\rangle |A^x \mod N\rangle
\]

3. We can equivalently express this state as
\[
\frac{1}{\sqrt{|M/r|}} \sum_{\ell = 0}^{[M/r]-1} |x_0 + \ell r\rangle |y_0\rangle
\]
where \( x_0 \) is the smallest number such that \( A^{x_0} = y_0 \mod N \). Measure the second register to obtain some \( y_0 \)

4. Apply the QFT to the first register
\[
\frac{1}{\sqrt{M \sqrt{|M/r|}}} \sum_{x \in \mathbb{Z}_M} \sum_{\ell = 0}^{[M/r]-1} \omega_M^{(x_0 + \ell r)x} |x\rangle |y_0\rangle
\]

5. Measure the first register to obtain some \( x \in \mathbb{Z}_M \). Find the best rational approximation \( \frac{a}{b} \) (with \( \gcd(a, b) = 1 \)) for the fraction \( \frac{x}{M} \) with denominator \( b \) at most \( 40M \). If \( A^b = A \mod M \) then output \( b. \)

**Shor’s Algorithm** [1]

Now we can describe Shor’s algorithm. Using state of the art multiplication algorithms, this runs in time \( \mathcal{O}(\log^2 N \log(\log N)) \) [12].

1. Randomly pick a number \( 1 < A < N \)
2. Compute \( \gcd(A, N) \). If this is not 1, then we are done.
3. Compute the period \( r \) of \( A \mod N \) using the order finding algorithm described above
4. If \( r \) is odd return to step 1
5. If \( a^{r/2} = -1 \mod N \) return to step 1
6. At this point, at least one of \( a^{r/2} - 1 \) and \( a^{r/2} + 1 \) is a nontrivial factor of \( N \), so we are done.
1.3 Quantum Complexity

All of the above algorithms run in time polynomial in the input size, and so can be considered efficient. Similar to the definition of BPP, the class of all problems solvable in quantum polynomial time will be referred to as BQP, or Bounded-error Quantum Polynomial time. Formally, let $L$ be a language over $\{0, 1\}^*$. We say that $L \in \text{BQP}$ iff there is a uniform circuit $C$ whose depth on $n$ qubits is $\text{poly}(n)$, such that $C$ decides $L$. $C$ decides $L$ if, given an input $x \in L$ it will output 1 with probability $> \frac{2}{3}$, and given an input $x \notin L$ it will output 1 with probability $< \frac{1}{3}$.

We can also consider natural generalizations of other classical complexity classes, such as NP. Unfortunately, quantum computers are inherently random, so we need a randomized version of NP, which is usually referred to as MA (Merlin Arthur). A language $L \subset \{0, 1\}^*$ is in MA if there exists an algorithm $A \in \text{BPP}$ such that for any $x \in L$, there is a $y$ of size $\text{poly}(|x|)$ such that on input $(x, y) A$ will output 1 with probability at least $\frac{2}{3}$. For any $x \notin L$, there is no $y$ of size $\text{poly}(|x|)$ such that on input $(x, y) A$ will output 1 with probability more than $\frac{1}{3}$. The analogous quantum complexity class, QMA (quantum merlin arthur) is precisely analogous except with $A \in \text{BQP}$ and $y$ being a quantum state on $\text{poly}(|x|)$ registers.

The notion of QMA-hardness and QMA-completeness are as expected; a problem $R$ is QMA-hard if every problem in QMA reduces to $R$ in polynomial deterministic time. A QMA-hard problem $R$ is QMA-complete if $R \in \text{QMA}$. The canonical example of a QMA-complete problem is the $k$-local hamiltonian. We are given a "hamiltonian", which is a $2^n \times 2^n$ hermitian matrix of the form

$$H = \sum_i h_i$$

(14)

Where each $h_i$ acts on at most $k$ qubits. We can consider, for example, the 5-local hamiltonian in which each $h_i$ can be described by a $2^5 \times 2^5$ matrix, and there are at most $\binom{n}{5}$ terms. We are given an $a, b; 0 \leq a, b - a \geq \text{poly}^{-1}(n)$, and asked to decide which statement is true, promised that one is:

1. Every eigenvalue of $H$ is above $b$
2. There is an eigenvalue of $H$ below $a$

This problem can be seen as a generalization of 5-sat. To see the correspondence, consider a clause like

$$C = (x_a \land \neg x_b \land x_c \land x_d \land \neg x_e)$$

(15)

Where these are the first 5 variables. If we include a term in our hamiltonian which looks like

$$h_C = |01001\rangle \langle 01001| \otimes I$$

(16)

Then we see that any eigenstate which assigns $x_a, ..., x_e$ such that the first clause will not be satisfied has eigenvalue $+1$ from this term. If we do this for all clauses, then we are left with a hamiltonian which has a 0 eigenvalue if and only if the original formula is satisfiable. However, the language of hamiltonians is more general, as it allows for more complicated $h_i$ functions.

Proving that this problem is in QMA is fairly simple: we call the $y$ in the definition of QMA the certificate. In case (2) above, we can simply provide the eigenstate $|\psi\rangle = y$ such that

$$\langle \psi | H | \psi \rangle < a$$

$$H | \psi \rangle \propto | \psi \rangle$$

(17)

(18)
This will convince a polynomial time verifier, which can compute the above condition, that we are in case (2). In case (1), there is no state which can be provided having eigenvalue less than \( a \), so our verifier cannot be convinced. It follows that 5-local Hamiltonian is in QMA. It remains to show that it is QMA-complete.

We will show this by simulating an arbitrary QMA computation [2]. In particular, we are given a circuit \( U = U_L...U_1 \) and a promise that either:

1. There is a \( |\psi\rangle \) such that \( U \) accepts \( |\psi\rangle \) with probability greater than \( 2/3 \)
2. For all \( |\psi\rangle \), \( U \) accepts \( |\psi\rangle \) with probability less than \( 1/3 \)

One can interpret \( U \) as having an \( x \) already hard wired, and we are simply trying to find a certificate. We want to construct a \( H \) such that \( H \) has a small eigenvalue in case (1) but not in case (2). To do so, intuitively we would like \( |\psi\rangle \) to be a 0 eigenvalue if it encodes a valid computation history of \( U \) that is accepting. We will express these constraints in a Hamiltonian as

\[
H = H_{\text{init}} + H_{\text{final}} + H_{\text{propagate}}
\]  

The eigenstate \( |\psi\rangle \) will be something like

\[
|\psi\rangle = \sum_t |\phi_t\rangle |t\rangle
\]

Where \( |t\rangle \) marks the computation step, running from 0 to \( l + 1 \), and \( |\phi_t\rangle \) marks the computation state. We would like \( H_{\text{init}} \) to penalize invalid starting states, e.g. any state that is not all 0’s on ancillas. This can simply be done by having

\[
H_{\text{init}} = \sum_{s > n} I^{s-1} \otimes |1\rangle \langle 1| \otimes I^{n-s}
\]  

Or a penalty for every non-zero entry in the ancillas. This is 1-local. Similarly, we penalize non-one end state in the final first qubit, using \( H_{\text{final}} \) of the form

\[
H_{\text{final}} = |0\rangle \langle 0| \otimes I
\]

Now it only remains to enforce in \( H_{\text{propagate}} \) that the history encoded by \( |\psi\rangle \) is a valid computation history. This can be done by ensuring that, at each time step, the corresponding gate is applied as given in \( U \). Each gate is at most 2-local, and there are 2 time-sites we want to check against (noting that the ”clock” register can be encoded in unary so that each step makes only a local change), meaning there is a polynomial set of 5-local operators which has eigenvalue 0 if and only if the step \( t \rightarrow t + 1 \) is valid. Putting these all into \( H \) and weighting each term with 1, we have constructed a Hamiltonian which has eigenvalue 0 if we are in case (1), and at least 1 if we are in case (2).

2 Quantum Max-Cut

This section will detail the problem of Quantum Max-Cut, as well as classical complexity results concerning it. The basics of quantum information from the previous section will be helpful, but the two are largely separate.
2.1 Classical Max-Cut

To begin, let us review the definition of classical Max-Cut. Suppose we have a weighted graph \( G = (V, E, w) \) where we enforce that the weights sum to 1 so that they form a probability distribution. A cut is a function, \( f : V \to \{1, -1\} \). The value of a cut is given by

\[
E \left[ \frac{1}{2} - \frac{1}{2} f(u)f(v) \right].
\] (23)

The Max-Cut problem asks for the value of the largest cut (See Figure 2 for an example.), i.e. the quantity

\[
\text{Max-Cut}(G) = \max_{f : V \to \{1, -1\}} E \left[ \frac{1}{2} - \frac{1}{2} f(u)f(v) \right].
\] (24)

Figure 2: An example of a Max-Cut instance. The max cut is given by the dotted line, and has a value 0.8.

We can physically interpret the Max-Cut problem as follows, which will justify the form of formula 24. Given our weighted graph, we now think of our vertices as classical bits. Then every bit string of length \( |V| \) will correspond to some cut of the graph. Formula 23 is now maximized when high weight edges are between opposite bits (or in other words, vertices have different cut values).

Max-Cut is NP-Hard to approximate better than a factor of \( \frac{16}{17} \). Goemans and Williamson gave an algorithm for Max-Cut using an SDP that achieved an approximation ratio of 0.878 [6], which was later shown to be optimal under the Unique Games Conjecture [7].

The Goemans and Williamson algorithm is relatively simple to state and is generalized in the study of Quantum Max-Cut. We first formulate a Semidefinite Program for Max-Cut:

\[
SDP_{MC}(G) = \max_{f : V \to S^{n-1}} E \left[ \frac{1}{2} - \frac{1}{2} f(u)f(v) \right]
\] (25)

where \( S^{n-1} \) is the hollow unit sphere in \( n = |V| \) dimensions (note that \( S^0 = \{1, -1\} \)). This is a valid relaxation because \( SDP_{MC}(G) \geq \text{Max-Cut}(G) \) for all graphs \( G \). It can be approximated in polynomial time, meaning that one can compute an optimal \( f \) with value \( SDP_{MC}(G) \pm \epsilon \) in time \( poly(n) \log(1/\epsilon) \). We then round the solution back to our domain through the following procedure (dubbed halfspace rounding):

1. Sample a random vector \( z = (z_1, z_2, \ldots, z_n) \) from the \( n \)-dimensional Gaussian distribution.
2. For each \( u \in V \), set \( f(u) \) to be
\[
\text{sgn}(\langle z, f_{SDP}(u) \rangle) = \frac{\langle z, f_{SDP}(u) \rangle}{|\langle z, f_{SDP}(u) \rangle|}
\] (26)

Goemans and Williamson then showed that for each \( u, v \in V \) that
\[
E_{(u,v) \sim E} \left[ \frac{1}{2} - \frac{1}{2} f(u)f(v) \right] \geq \alpha_{GW} \cdot \left[ \frac{1}{2} - \frac{1}{2} f_{SDP}(u)f_{SDP}(v) \right]
\] (27)
where \( \alpha_{GW} = 0.878567 \). Taking an average over the edges in \( G \), this shows that the expected value of \( f \) is at least \( \alpha_{GW} \cdot SDP_{MC}(G) \geq \alpha_{GW} \text{Max-Cut}(G) \). Thus, the Goemans and Williamson algorithm is an \( \alpha_{GW} \)-approximation algorithm.

### 2.2 Quantum Max-Cut

Under the bit assignment interpretation of Max-Cut, the obvious quantum analogue would be to assign a qubit to each vertex, and reward adjacent vertices with ”opposite” qubits. Two qubits are opposite if they are orthogonal states, e.g. not only |0\rangle, |1\rangle but also |0\rangle + |1\rangle, |0\rangle − |1\rangle. Formally, on a graph \( G = (V, E, w) \) who’s edge weights form a probability distribution, we could write
\[
H_G = \frac{1}{4}I - \frac{1}{4} \sum_{(u,v) \sim G} [X_uX_v + Y_uY_v + Z_uZ_v]
\] (28)
Intuitively, this rewards opposite-direction adjacent spins in all bases equally, meaning something like a bell pair which is entangled but anti-correlated in all bases (referred to in the literature as \(|\Phi_{11}\rangle\)) would be the state which maximizes the the two-qubit \( h \) term in \( H_G \). We note that this problem of finding the maximal quantum cut can be thought of as a special case of the general hamiltonian ground state problem discussed in section 1.3. We are maximizing instead of minimizing the ”energy”, but this can be changed by inverting the hamiltonian. We would like to say that this problem is not only NP-hard to solve exactly, but also to approximate better than some value.

We can consider making a guess for the form of the cut state \(|\psi\rangle\), which we call an ansatz. One such simple state is a product state, meaning one that can be written as
\[
|\psi\rangle = |q_0\rangle \otimes |q_1\rangle \otimes ... \otimes |q_n\rangle
\] (29)
Where each \( q_i \) are single qubit states. Such states can be efficiently represented classically, as there are only \( 2^n \) many amplitudes needed in the above expression. We note that there is a standard equivalence between points on \( S^2 \) and pure qubits, commonly referred to as the bloch sphere. Under this restriction to product states, one could therefore write the optimization problem to be
\[
\max_{f:V \to S^2} \left( \frac{1}{4} - \frac{1}{4} \langle f(u), f(v) \rangle \right)
\] (30)
A natural ”relaxation” of this problem, meaning a modification by adding degrees of freedom, which one could consider is as follows
\[
\max_{f:V \to S^{n-1}} \left( \frac{1}{4} - \frac{1}{4} \langle f(u), f(v) \rangle \right)
\] (31)
We note that, in contrast with eqn. 30, we have that eqn. 31 is a semi-definite program. It is a well known result in the literature that semi-definite programs admit efficient classical algorithms, so
this weaker optimization can be solved classically (similar to in the classical case). Given a function 
\( f : V \rightarrow S^{n-1} \), we can then construct a satisfying \( f' : V \rightarrow S^2 \) which hopefully approximates the 
true maximal value. This is done using a technique called projection rounding, and it proceeds as 
follows.

1. Sample a random \( 3 \times n \) matrix \( Z \), where each entry is i.i.d. gaussian

2. For every vertex \( v \in V \), assign 
\[
f'(v) = Z f(v) / \|Z f(v)\|_2
\]

This procedure turns out to be provably optimal \([3, 10]\), in the sense that it is a 0.956 approximation 
to the fully optimal product state solution, and under the unique games conjecture it is impossible 
to do better than this in polynomial time.

Now we explain a few results for the general Quantum Max-Cut problem, where we no longer 
restrict ourselves to product state solutions. With some nontrivial work, \([10]\) derive a basic SDP 
relaxtion for general Quantum Max-Cut.

\[
\max_{f : V \rightarrow S^{n-1}} \left( \mathbb{E}_{(u,v) \sim G} \left[ \frac{1}{4} - \frac{3}{4} \langle f(u), f(v) \rangle \right] \right)
\]

Gharibian and Parekh in \([5]\) apply projection rounding on the solution to this SDP to achieve 
an \( \alpha_{GP} = 0.498 \) approximation algorithm for Quantum Max-Cut. In fact, it turns out that this 
is the optimal rounding strategy to use for the Quantum Max-Cut SDP. Separately, \([4]\) give a 
0.53 approximation algorithm for Quantum Max-Cut using other techniques. Putting these two 
pieces of information together shows that relaxing the basic SDP is not the optimal way to solve the 
problem. This is significant because the basic SDP solution is optimal for both classical Max-Cut 
and the product state version of Quantum Max-Cut.

3 Open Problems

We begin by detailing some open problems related to Quantum Max-Cut \([10]\)

1. A potentially unintuitive result about quantum max-cut is that the hard-to-approximate 
instances are ones where the graph is fairly sparse \([9]\). Not many explicit families of hard 
instances are known \([7]\), and finding more could prove interesting.

2. Is there a hardness of approximation result for Quantum Max-Cut which improves on these 
bounds?

3. Is there a better approximation result for Quantum Max-Cut, e.g. a better algorithm?

We began looking for algorithms in the category of \((4)\). A natural way of doing this is by trying 
to find ansatz which are able to capture more complicated features of general quantum states. 
This takes inspiration from previous work by Anshu et. al. \([4]\), which was able to improve on 
the one-qubit approximation upper bound of 0.498 \([5]\) by using a two-qubit general ansatz. In 
particular, the quantum state is composed of an arbitrary product of two-qubit states, which allows 
them to achieve an approximation ratio of 0.53. A natural continuation of this project would be to 
attempt a matrix product state ansatz. A matrix product state is a state \( |\psi\rangle \) which can be written 
as the product of many matrices \( A, B, ..., K \), where \( A \) has a single row and \( K \) has a single column. 
The entries in these matrices are kets, such that this product evaluates to a single ket being \( |\psi\rangle \). 
Everything is a matrix product state, but we will consider states which have a polynomial number 
of matrix entries so they can be efficiently represented. Pictorially, we could visualize these as
Unfortunately, the natural relaxation SDP is no longer applicable, as it is non-obvious how to round to a matrix product state. This is because the natural relaxation treats each vertex as having an independent value, thereby seeming to not capture entanglement. This is reflected in the fact that the two-qubit ansatz needed a different SDP \cite{4}. A potential path around this limitation is imposing some structure on the matrix, e.g. the dimension and number; however this does not entirely solve the problem, as a MPS needs to spit out a correctly normalized state. It is unclear how to express these constraints as a semi-definite program, as it is more complicated than any specific matrix being PSD. In this way, we have identified natural barriers to an attempt to produce a better approximation algorithm by using matrix product states as an ansatz, and then relaxing to an SDP. It may be possible to proceed by forgoing the usage of an SDP and attempting a direct approximation, but we did not see a feasible way.

\[
|\psi\rangle = [|\alpha_{11}\rangle \ldots |\alpha_{1n_1}\rangle] \begin{bmatrix} |\beta_{11}\rangle & \ldots & |\beta_{1n_2}\rangle \\ \vdots & \ddots & \vdots \\ |\beta_{n_11}\rangle & \ldots & |\beta_{n_1n_2}\rangle \end{bmatrix} \begin{bmatrix} |\kappa_{11}\rangle \\ \vdots \\ |\kappa_{n_{d1}}\rangle \end{bmatrix}
\]
References


