MIT-CTP/4610

A Quantum Approximate Optimization Algorithm

Edward Farhi and Jeffrey Goldstone

Center for Theoretical Physics

Massachusetts Institute of Technology

Cambridge, MA 02139

Sam Gutmann

Abstract

We introduce a quantum algorithm that produces approximate solutions for combinatorial optimization problems. The algorithm depends on an integer $p \geq 1$ and the quality of the approximation improves as p is increased. The quantum circuit that implements the algorithm consists of unitary gates whose locality is at most the locality of the objective function whose optimum is sought. The depth of the circuit grows linearly with p times (at worst) the number of constraints. If p is fixed, that is, independent of the input size, the algorithm makes use of efficient classical preprocessing. If p grows with the input size a different strategy is proposed. We study the algorithm as applied to MaxCut on regular graphs and analyze its performance on 2-regular and 3-regular graphs for fixed p. For $p = 1$, on 3-regular graphs the quantum algorithm always finds a cut that is at least 0.6924 times the size of the optimal cut.

I. INTRODUCTION

Combinatorial optimization problems are specified by n bits and m clauses. Each clause is a constraint on a subset of the bits which is satisfied for certain assignments of those bits and unsatisfied for the other assignments. The objective function, defined on n bit strings, is the number of satisfied clauses,

$$C(z) = \sum_{\alpha=1}^{m} C_\alpha(z) \tag{1}$$

where $z = z_1 z_2 \dots z_n$ is the bit string and $C_\alpha(z) = 1$ if z satisfies clause α and 0 otherwise. Typically C_α depends on only a few of the n bits. Satisfiability asks if there is a string that satisfies every clause. MaxSat asks for a string that maximizes the objective function. Approximate optimization asks for a string z for which $C(z)$ is close to the maximum of C. In this paper we present a general quantum algorithm for approximate optimization. We study its performance in special cases of MaxCut and also propose an alternate form of the algorithm geared toward finding a large independent set of vertices of a graph.

The quantum computer works in a 2^n dimensional Hilbert space with computational basis vectors $|z\rangle$, and we view (1) as an operator which is diagonal in the computational basis. Define a unitary operator $U(C, \gamma)$ which depends on an angle γ,

$$U(C,\gamma) = e^{-i\gamma C} = \prod_{\alpha=1}^{m} e^{-i\gamma C_\alpha} . \tag{2}$$

All of the terms in this product commute because they are diagonal in the computational basis and each term's locality is the locality of the clause α. Because C has integer eigenvalues we can restrict γ to lie between 0 and 2π. Define the operator B which is the sum of all single bit σ^x operators,

$$B = \sum_{j=1}^{n} \sigma_j^x . \tag{3}$$

Now define the β dependent product of commuting one bit operators

$$U(B,\beta) = e^{-i\beta B} = \prod_{j=1}^{n} e^{-i\beta\sigma_j^x} \tag{4}$$

where β runs from 0 to π. The initial state $|s\rangle$ will be the uniform superposition over computational basis states:

$$|s\rangle = \frac{1}{\sqrt{2^n}} \sum_z |z\rangle . \tag{5}$$

For any integer $p \geq 1$ and $2p$ angles $\gamma_1 \ldots \gamma_p \equiv \boldsymbol{\gamma}$ and $\beta_1 \ldots \beta_p \equiv \boldsymbol{\beta}$ we define the angle dependent quantum state:

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = U(B, \beta_p)\, U(C, \gamma_p) \cdots U(B, \beta_1)\, U(C, \gamma_1)\, |s\rangle\,. \tag{6}$$

Even without taking advantage of the structure of the instance, this state can be produced by a quantum circuit of depth at most $mp + p$. Let F_p be the expectation of C in this state

$$F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \boldsymbol{\gamma}, \boldsymbol{\beta}|\, C\, |\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle\,. \tag{7}$$

and let M_p be the maximum of F_p over the angles,

$$M_p = \max_{\boldsymbol{\gamma}, \boldsymbol{\beta}} F_p\,(\boldsymbol{\gamma}, \boldsymbol{\beta}). \tag{8}$$

Note that the maximization at $p-1$ can be viewed as a constrained maximization at p so

$$M_p \geq M_{p-1}. \tag{9}$$

Furthermore we will later show that

$$\lim_{p \to \infty} M_p = \max_z C(z). \tag{10}$$

These results suggest a way to design an algorithm. Pick a p and start with a set of angles $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ that somehow make F_p as large as possible. Use the quantum computer to get the state $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$. Measure in the computational basis to get a string z and evaluate $C(z)$. Repeat with the same angles. Enough repetitions will produce a string z with $C(z)$ very near or greater than $F_p(\boldsymbol{\gamma}, \boldsymbol{\beta})$. The rub is that it is not obvious in advance how to pick good angles.

If p doesn't grow with n, one possibility is to run the quantum computer with angles $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ chosen from a fine grid on the compact set $[0, 2\pi]^p \times [0, \pi]^p$, moving through the grid to find the maximum of F_p. Since the partial derivatives of $F_p(\boldsymbol{\gamma}, \boldsymbol{\beta})$ in (7) are bounded by $\mathcal{O}(m^2 + mn)$ this search will efficiently produce a string z for which $C(z)$ is close to M_p or larger. However we show in the next section that if p does not grow with n and each bit is involved in no more than a fixed number of clauses, then there is an efficient classical calculation that determines the angles that maximize F_p. These angles are then used to run the quantum computer to produce the state $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$ which is measured in the computational basis to get a string z. The mean of $C(z)$ for strings obtained in this way is M_p.

II. FIXED p ALGORITHM

We now explain how for fixed p we can do classical preprocessing and determine the angles $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ that maximize $F_p(\boldsymbol{\gamma}, \boldsymbol{\beta})$. This approach will work more generally but we illustrate it for a specific problem, MaxCut for graphs with bounded degree. The input is a graph with n vertices and an edge set $\{\langle jk \rangle\}$ of size m. The goal is to find a string z that makes

$$C = \sum_{\langle jk \rangle} C_{\langle jk \rangle}, \tag{11}$$

where

$$C_{\langle jk \rangle} = \frac{1}{2} \left(-\sigma_j^z \sigma_k^z + 1 \right), \tag{12}$$

as large as possible. Now

$$F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \sum_{\langle jk \rangle} \langle s | \, U^\dagger(C, \gamma_1) \cdots U^\dagger(B, \beta_p) \, C_{\langle jk \rangle} U(B, \beta_p) \cdots U(C, \gamma_1) \, | s \rangle \, . \tag{13}$$

Consider the operator associated with edge $\langle jk \rangle$

$$U^\dagger(C, \gamma_1) \cdots U^\dagger(B, \beta_p) C_{\langle jk \rangle} U(B, \beta_p) \cdots U(C, \gamma_1). \tag{14}$$

This operator only involves qubits j and k and those qubits whose distance on the graph from j or k is less than or equal to p. To see this consider $p = 1$ where the previous expression is

$$U^\dagger(C, \gamma_1) \, U^\dagger(B, \beta_1) C_{\langle jk \rangle} U(B, \beta_1) \, U(C, \gamma_1). \tag{15}$$

The factors in the operator $U(B, \beta_1)$ which do not involve qubits j or k commute through $C_{\langle jk \rangle}$ and we get

$$U^\dagger(C, \gamma_1) \, e^{i\beta_1(\sigma_j^x + \sigma_k^x)} C_{\langle jk \rangle} e^{-i\beta_1(\sigma_j^x + \sigma_k^x)} \, U(C, \gamma_1). \tag{16}$$

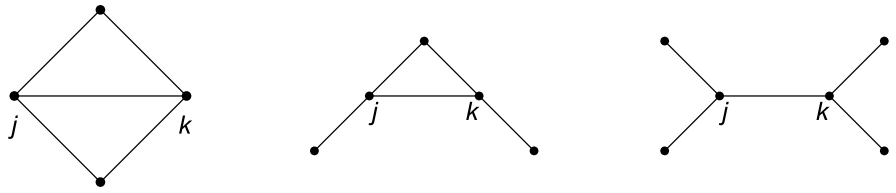
Any factors in the operator $U(C, \gamma_1)$ which do not involve qubits j or k will commute through and cancel out. So the operator in equation (16) only involves the edge $\langle jk \rangle$ and edges adjacent to $\langle jk \rangle$, and qubits on those edges. For any p we see that the operator in (14) only involves edges at most p steps away from $\langle jk \rangle$ and qubits on those edges.

Return to equation (13) and note that the state $|s\rangle$ is the product of σ^x eigenstates

$$|s\rangle = |+\rangle_1 \, |+\rangle_2 \ldots |+\rangle_n \tag{17}$$

so each term in equation (13) depends only on the subgraph involving qubits j and k and those at a distance no more than p away. These subgraphs each contain a number of qubits that is independent of n (because the degree is bounded) and this allows us to evaluate F_p in terms of quantum subsystems whose sizes are independent of n.

As an illustration consider MaxCut restricted to input graphs of fixed degree 3. For $p = 1$, there are only these possible subgraphs for the edge $\langle jk \rangle$:



(18)

We will return to this case later.

For any subgraph G define the operator C_G which is C restricted to G,

$$C_G = \sum_{\langle \ell \ell' \rangle \epsilon G} C_{\langle \ell \ell' \rangle}, \tag{19}$$

and the associated operator

$$U(C_G, \gamma) = e^{-i\gamma C_G}. \tag{20}$$

Also define

$$B_G = \sum_{j \epsilon G} \sigma_j^x \tag{21}$$

and

$$U(B_G, \beta) = e^{-i\beta B_G}. \tag{22}$$

Let the state $|s, G\rangle$ be

$$|s, G\rangle = \prod_{\ell \epsilon G} |+\rangle_\ell .$$

Return to equation (13). Each edge $\langle j, k \rangle$ in the sum is associated with a subgraph $g(j, k)$ and makes a contribution to F_p of

$$\langle s, g(j,k)| U^\dagger \left(C_{g(j,k)}, \gamma_p\right) \cdots U^\dagger (B_{g(j,k)}, \beta_1) C_{\langle jk \rangle} U \left(B_{g(j,k)}, \beta_1\right) \cdots U \left(C_{g(j,k)}, \gamma_p\right) |s, g(j,k)\rangle \tag{23}$$

The sum in (13) is over all edges, but if two edges $\langle jk \rangle$ and $\langle j'k' \rangle$ give rise to isomorphic subgraphs, then the corresponding functions of $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ are the same. Therefore we can view

the sum in (13) as a sum over subgraph types. Define

$$f_g(\boldsymbol{\gamma},\boldsymbol{\beta}) = \langle s, g(j,k)|\, U^\dagger(C_{g(j,k)},\gamma_1)\cdots U^\dagger(B_{g(j,k)},\beta_p) C_{\langle jk\rangle} U(B_{g(j,x)}\beta_p)\cdots \\ U(C_{g(j,k)},\gamma_1)\,|s,g(j,k)\rangle\,, \quad (24)$$

where $g(j,k)$ is a subgraph of type g. F_p is then

$$F_p(\boldsymbol{\gamma},\boldsymbol{\beta}) = \sum_g w_g\, f_g(\boldsymbol{\gamma},\boldsymbol{\beta}) \quad (25)$$

where w_g is the number of occurrences of the subgraph g in the original edge sum. The functions f_g do not depend on n and m. The only dependence on n and m comes through the weights w_g and these are just read off the original graph. Note that the expectation in (24) only involves the qubits in subgraph type g. The maximum number of qubits that can appear in (23) comes when the subgraph is a tree. For a graph with maximum degree v, the numbers of qubits in this tree is

$$q_{\text{tree}} = 2\left[\frac{(v-1)^{p+1}-1}{(v-1)-1}\right], \quad (26)$$

(or $2p+2$ if $v=2$), which is n and m independent. For each p there are only finitely many subgraph types.

Using (24), $F_p(\boldsymbol{\gamma},\boldsymbol{\beta})$ in (25) can be evaluated on a classical computer whose resources are not growing with n. Each f_g involves operators and states in a Hilbert space whose dimension is at most $2^{q_{\text{tree}}}$. Admittedly for large p this may be beyond current classical technology, but the resource requirements do not grow with n.

To run the quantum algorithm we first find the $(\boldsymbol{\gamma},\boldsymbol{\beta})$ that maximize F_p. The only dependence on n and m is in the weights w_g and these are easily evaluated. Given the best $(\boldsymbol{\gamma},\boldsymbol{\beta})$ we turn to the quantum computer and produce the state $|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle$ given in equation (6). We then measure in the computational basis and get a string z and evaluate $C(z)$. Repeating gives a sample of values of $C(z)$ between 0 and $+m$ whose mean is $F_p(\boldsymbol{\gamma},\boldsymbol{\beta})$. An outcome of at least $F_p(\boldsymbol{\gamma},\boldsymbol{\beta})-1$ will be obtained with probability $1-1/m$ with order $m\log m$ repetitions.

III. CONCENTRATION

Still using MaxCut on regular graphs as our example, it is useful to get information about the spread of C measured in the state $|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle$. If v is fixed and p is fixed (or grows

slowly with n) the distribution of $C(z)$ is actually concentrated near its mean. To see this, calculate

$$\langle \boldsymbol{\gamma}, \boldsymbol{\beta} | \, C^2 \, | \boldsymbol{\gamma}, \boldsymbol{\beta} \rangle - \langle \boldsymbol{\gamma}, \boldsymbol{\beta} | \, C \, | \boldsymbol{\gamma}, \boldsymbol{\beta} \rangle^2 \tag{27}$$

$$\begin{aligned} = \sum_{\substack{\langle jk \rangle \\ \langle j'k' \rangle}} \Big[& \langle s | \, U^\dagger(C, \gamma_1) \cdots U^\dagger(B, \beta_p) \, C_{\langle jk \rangle} \, C_{\langle j'k' \rangle} \, U(B, \beta_p) \cdots U(C, \gamma_1) \, | s \rangle \\ & - \langle s | \, U^\dagger(C, \gamma_1) \cdots U^\dagger(B, \beta_p) \, C_{\langle jk \rangle} \, U(B, \beta_p) \cdots U(C, \gamma_1) \, | s \rangle \\ & \cdot \langle s | \, U^\dagger(C, \gamma_1) \cdots U^\dagger(B, \beta_p) \, C_{\langle j'k' \rangle} \, U(B, \beta_p) \cdots U(C, \gamma_1) \, | s \rangle \Big]. \end{aligned} \tag{28}$$

If the subgraphs $g(j,k)$ and $g(j',k')$ do not involve any common qubits, the summand in (28) will be 0. The subgraphs $g(j,k)$ and $g(j',k')$ will have no common qubits as long as there is no path in the instance graph from $\langle jk \rangle$ to $\langle j'k' \rangle$ of length $2p+1$ or shorter. From (26) with p replaced by $2p+1$ we see that for each $\langle jk \rangle$ there are at most

$$2 \left[\frac{(v-1)^{2p+2} - 1}{(v-1) - 1} \right] \tag{29}$$

edges $\langle j'k' \rangle$ which could contribute to the sum in (28) (or $4p+4$ if $v = 2$) and therefore

$$\langle \boldsymbol{\gamma}, \boldsymbol{\beta} | \, C^2 \, | \boldsymbol{\gamma}, \boldsymbol{\beta} \rangle - \langle \boldsymbol{\gamma}, \boldsymbol{\beta} | \, C \, | \boldsymbol{\gamma}, \boldsymbol{\beta} \rangle^2 \leqslant 2 \left[\frac{(v-1)^{2p+2} - 1}{(v-1) - 1} \right] \cdot m \tag{30}$$

since each summand is at most 1 in norm. For v and p fixed we see that the standard deviation of $C(z)$ is at most of order $\sqrt{m}$. This implies that the sample mean of order m^2 values of $C(z)$ will be within 1 of $F_p(\boldsymbol{\gamma}, \boldsymbol{\beta})$ with probability $1 - \frac{1}{m}$. The concentration of the distribution of $C(z)$ also means that there is only a small probability that the algorithm will produce strings with $C(z)$ much bigger than $F_p(\boldsymbol{\gamma}, \boldsymbol{\beta})$.

IV. THE RING OF DISAGREES

We now analyze the performance of the quantum algorithm for MaxCut on 2-regular graphs. Regular of degree 2 (and connected) means that the graph is a ring. The objective operator is again given by equation (11) and its maximum is n or $n-1$ depending on whether n is even or odd. We will analyze the algorithm for all p.

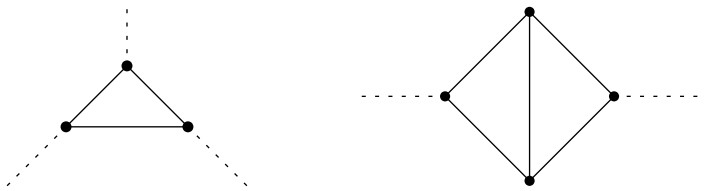
For any p (less than $n/2$), for each edge in the ring, the subgraph of vertices within p of the edge is a segment of $2p+2$ connected vertices with the given edge in the middle. So for

each p there is only one type of subgraph, a line segment of $2p+2$ qubits and the weight for this subgraph type is n. We numerically maximize the function given in (24) and we find that for $p = 1, 2, 3, 4, 5$ and 6 the maxima are 3/4, 5/6, 7/8, 9/10, 11/12, and 13/14 to 13 decimal places from which we conclude that $M_p = n(2p+1)/(2p+2)$ for all p. So the quantum algorithm will find a cut of size $n(2p+1)/(2p+2) - 1$ or bigger. Since the best cut is n, we see that our quantum algorithm can produce an approximation ratio that can be made arbitrarily close to 1 by making p large enough, independent of n. For each p the circuit depth can be made $3p$ by breaking the edge sum in C into two sums over $\langle j, j+1\rangle$ with j even and j odd. So this algorithm has a circuit depth independent of n.

V. MAXCUT ON 3-REGULAR GRAPHS

We now look at how the Quantum Approximate Optimization Algorithm, the QAOA, performs on MaxCut on (connected) 3-regular graphs. The approximation ratio is $C(z)$, where z is the output of the quantum algorithm, divided by the maximum of C. We first show that for $p = 1$, the worst case approximation ratio that the quantum algorithm produces is 0.6924.

Suppose a 3-regular graph with n vertices (and accordingly $3n/2$ edges) contains T "isolated triangles" and S "crossed squares", which are subgraphs of the form,



. (31)

The dotted lines indicate edges that leave the isolated triangle and the crossed square. To say that the triangle is isolated is to say that the 3 edges that leave the triangle end on distinct vertices. If the two edges that leave the crossed square are in fact the same edge, then we have a 4 vertex disconnected 3-regular graph. For this special case (the only case where the analysis below does not apply) the approximation ratio is actually higher than 0.6924. In general, $3T + 4S \leq n$ because no isolated triangle and crossed square can share a vertex.

Return to the edge sum in $F_1(\gamma, \beta)$ of equation (13). For each crossed square there is

one edge $\langle jk \rangle$ for which $g(j,k)$ is the first type displayed in (18). Call this subgraph type g_4 because it has 4 vertices. In each crossed square there are 4 edges that give rise to subgraphs of the second type displayed in (18). We call this subgraph type g_5 because it has 5 vertices. All 3 of the edges in any isolated triangle have subgraph type g_5, so there are $4S+3T$ edges with subgraph type g_5. The remaining edges in the graph all have a subgraph type like the third one displayed in (18) and we call this subgraph type g_6. There are $(3n/2 - 5S - 3T)$ of these so we have

$$F_1(\gamma,\beta) = S f_{g_4}(\gamma,\beta) + (4S+3T)\, f_{g_5}(\gamma,\beta) + \left(\frac{3n}{2} - 5S - 3T\right) f_{g_6}(\gamma,\beta) \tag{32}$$

The maximum of F_1 is a function of n, S, and T,

$$M_1(n,S,T) = \max_{\gamma,\beta} F_1(\gamma,\beta). \tag{33}$$

Given any 3 regular graph it is straightforward to count S and T. Then using a classical computer it is straightforward to calculate $M_1(n,S,T)$. Running a quantum computer with the maximizing angles γ and β will produce the state $|\gamma,\beta\rangle$ which is then measured in the computational basis. With order $n \log n$ repetitions a string will be found whose cut value is very near or larger than $M_1(n,S,T)$.

To get the approximation ratio we need to know the best cut that can be obtained for the input graph. This is not just a function of S and T. However a graph with S crossed squares and T isolated triangles must have at least one unsatisfied edge per crossed square and one unsatisfied edge per isolated triangle so the number of satisfied edges is $\leq (3n/2 - S - T)$. This means that for any graph characterized by n, S and T the quantum algorithm will produce an approximation ratio that is at least

$$\frac{M_1(n,S,T)}{\left(\frac{3n}{2} - S - T\right)}. \tag{34}$$

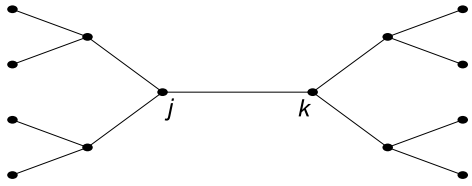
It is convenient to scale out n from the top and bottom of (34). Note that M_1/n which comes from F_1/n depends only on $S/n \equiv s$ and $T/n \equiv t$. So we can write (34) as

$$\frac{M_1(1,s,t)}{\left(\frac{3}{2} - s - t\right)} \tag{35}$$

where $s,t \geq 0$ and $4s+3t \leq 1$. It is straightforward to numerically evaluate (35) and we find that it achieves its minimum value at $s = t = 0$ and the value is 0.6924. So we know that on any 3-regular graph, the QAOA will always produce a cut whose size is at least 0.6924

times the size of the optimal cut. This $p = 1$ result on 3-regular graphs is not as good as known classical algorithms [1].

It is possible to analyze the performance of the QAOA for $p = 2$ on 3-regular graphs. However it is more complicated then the $p = 1$ case and we will just show partial results. The subgraph type with the most qubits is this tree with 14 vertices:



(36)

Numerically maximizing (24) with g given by (36) yields 0.7559. Consider a 3-regular graph on n vertices with $o(n)$ pentagons, squares and triangles. Then all but $o(n)$ edges have (36) as their subgraph type. The QAOA at $p = 2$ cannot detect whether the graph is bipartite, that is, completely satisfiable, or contains many odd loops of length 7 or longer. If the graph is bipartite the approximation ratio is 0.7559 in the limit of large n. If the graph contains many odd loops (length 7 or more), the approximation ratio will be higher.

VI. RELATION TO THE QUANTUM ADIABATIC ALGORITHM

We are focused on finding a good approximate solution to an optimization problem whereas the Quantum Adiabatic Algorithm, QAA [2], is designed to find the optimal solution and will do so if the run time is long enough. Consider the time dependent Hamiltonian $H(t) = (1 - t/T)B + (t/T)C$. Note that the state $|s\rangle$ is the highest energy eigenstate of B and we are seeking a high energy eigenstate of C. Starting in $|s\rangle$ we could run the quantum adiabatic algorithm and if the run time T were long enough we would find the highest energy eigenstate of C. Because B has only non-negative off-diagonal elements, the Perron-Frobenius theorem implies that the difference in energies between the top state and the one below is greater than 0 for all $t < T$, so for sufficiently large T success is assured. A Trotterized approximation to the evolution consists of an alternation of the operators $U(C, \gamma)$ and $U(B, \beta)$ where the sum of the angles is the total run time. For a good approximation we want each γ and β to be small and for success we want a long run time so together these

force p to be large. In other words, we can always find a p and a set of angles $\boldsymbol{\gamma}, \boldsymbol{\beta}$ that make $F_p(\boldsymbol{\gamma}, \boldsymbol{\beta})$ as close to M_p as desired. With (9), this proves the assertion of (10).

The previous discussion shows that we can get a good approximate solution to an optimization problem by making p sufficiently large, perhaps exponentially large in n. But the QAA works by producing a state with a large overlap with the optimal string. In this sense (10), although correct, may be misleading. In fact on the ring of disagrees the state produced at $p = 1$, which gives a 3/4 approximation ratio, has an exponentially small overlap with the optimal strings.

We also know an example where the QAA fails and the QAOA succeeds. In this example (actually a minimization) the objective function is symmetric in the n bits and therefore depends only on the Hamming weight. The objective function is plotted in figure 1 of reference [3]. Since the beginning Hamiltonian is also symmetric the evolution takes place in a subspace of dimension $n+1$ with a basis of states $|w\rangle$ indexed by the Hamming weight. The example can be simulated and analyzed for large n. For subexponential run times, the QAA is trapped in a false minimum at $w = n$. The QAOA can be similarly simulated and analyzed. For large n, even with $p = 1$, there are values of γ_1 and β_1 such that the final state is concentrated near the true minimum at $w = 0$.

The Quantum Approximate Optimization Algorithm has the key feature that as p increases the approximation improves. We contrast this to the performance of the QAA. For realizations of the QAA there is a total run time T that also appears in the instantaneous Hamiltonian, $H(t) = \widetilde{H}(t/T)$. We start in the ground state of $\widetilde{H}(0)$ seeking the ground state of $\widetilde{H}(1)$. As T goes to infinity the overlap of the evolved state with the desired state goes to 1. However the success probability is generally not a monotonic function of T. See figure 2 of reference [4] for an extreme example where the success probability is plotted as a function of T for a particular 20 qubit instance of Max2Sat. The probability rises and then drops dramatically, and the ultimate rise for large T is not seen for times that can be reasonably simulated. It may well be advantageous in designing strategies for the QAOA to use the fact that the approximation improves as p increases.

VII. A VARIANT OF THE ALGORITHM

We are now going to give a variant of the basic algorithm which is suited to situations where the search space is a complicated subset of the n bit strings. We work with an example that illustrates the basic idea. Consider the problem of finding a large independent set in a given graph of n vertices. An independent set is a subset of the vertices with the property that no two vertices in the subset have an edge between them. With the vertices labeled 1 to n, a subset of the vertices corresponds to the string $z = z_1 z_2 \dots z_n$ with each bit being 1 if the corresponding vertex is in the subset and the bit is 0 if the vertex is not. We restrict to strings which correspond to independent sets in the graph. The size of the independent set is the Hamming weight of the string z which we denote by $C(z)$,

$$C(z) = \sum_{j=1}^{n} z_j, \tag{37}$$

and the goal is to find a string z that makes $C(z)$ large.

The Hilbert space for our quantum algorithm has an orthonormal basis $|z\rangle$ where z is any string corresponding to an independent set. In cases of interest, the Hilbert space dimension is exponentially large in n, though not as big as 2^n. The Hilbert space is not a simple tensor product of qubits. The operator C is associated with the γ dependent unitary

$$U(C, \gamma) = e^{-i\gamma C} \tag{38}$$

where γ lies between 0 and 2π because C has integer eigenvalues. We define the quantum operator B that connects the basis states:

$$\langle z|B|z'\rangle = \begin{cases} 1 : z \text{ and } z' \text{ differ in one bit} \\ 0 : \text{otherwise} \end{cases} . \tag{39}$$

Note that B is the adjacency matrix of the hypercube restricted to the legal strings, that is, those that correspond to independent sets in the given graph. Now, in general, B does not have integer eigenvalues so we define

$$U(B, b) = e^{-ibB} \tag{40}$$

where b is a real number.

For the starting state of our algorithm we take the easy to construct state $|z=0\rangle$ corresponding to the empty independent set which has the minimum value of C. For $p \geq 1$, we have p real numbers $b_1, b_2 \ldots b_p \equiv \mathbf{b}$ and $p-1$ angles $\gamma_1, \gamma_2, .\gamma_{p-1} \equiv \boldsymbol{\gamma}$. The quantum state

$$|\mathbf{b}, \boldsymbol{\gamma}\rangle = U(B, b_p)U(C, \gamma_{p-1}) \cdots U(B, b_1)|z=0\rangle \tag{41}$$

is what we get after the application of an alternation of the operators associated with B and C. Now we can define

$$F_p(\mathbf{b}, \boldsymbol{\gamma}) = \langle \mathbf{b}, \boldsymbol{\gamma}|C|\mathbf{b}, \boldsymbol{\gamma}\rangle \tag{42}$$

as the expectation of C in the state $|\mathbf{b}, \boldsymbol{\gamma}\rangle$. And finally we define the maximum,

$$M_p = \max_{\mathbf{b}, \boldsymbol{\gamma}} F_p(\mathbf{b}, \boldsymbol{\gamma}). \tag{43}$$

The maximization at $p-1$ is the maximization at p with $b_p = 0$ and $\gamma_{p-1} = 0$ so we have

$$M_p \geq M_{p-1}. \tag{44}$$

Furthermore,

$$\lim_{p \to \infty} M_p = \max_{z \text{ legal}} C(z). \tag{45}$$

To see why (45) is true note that the initial state is the ground state of C, which we view as the state with the maximum eigenvalue of $-C$. We are trying to reach a state which is an eigenstate of $+C$ with maximum eigenvalue. There is an adiabatic path (which stays at the top of the spectrum throughout) with run time T that achieves this as T goes to infinity. This path has two parts. In the first we interpolate between the beginning Hamiltonian $-C$ and the Hamiltonian B,

$$H(t) = \left(1 - \frac{2t}{T}\right)(-C) + \frac{2t}{T}B \quad , \quad 0 \leq t \leq \frac{T}{2} \tag{46}$$

We evolve the initial state with this Hamiltonian for time $T/2$ ending arbitrarily close to the top state of B. Next we interpolate between the Hamiltonian B and the Hamiltonian $+C$,

$$H(t) = \left(2 - \frac{2t}{T}\right)B + \left(\frac{2t}{T} - 1\right)C \quad , \quad \frac{T}{2} \leq t \leq T \tag{47}$$

evolving the quantum state just produced from time $t = T/2$ to $t = T$. As in section VI, using the Perron-Frobenius Theorem, the Adiabatic Theorem and Trotterization we get the result given in (45).

Together (41) through (45) suggest a quantum subroutine for an independent set algorithm. For a given p and a given $(\mathbf{b}, \boldsymbol{\gamma})$ produce the quantum state $|\mathbf{b}, \boldsymbol{\gamma}\rangle$ of (41). Measure in the computational basis to get a string z which labels an independent set whose size is the Hamming weight of z. Repeat with the same $(\mathbf{b}, \boldsymbol{\gamma})$ to get an estimate of $F_p(\mathbf{b}, \boldsymbol{\gamma})$ in (42). This subroutine can be called by a program whose goal is to get close to M_p given by (43). This enveloping program can be designed using either the methods outlined in this paper or novel techniques.

For $p = 1$, the subroutine can be thought of as evolving the initial state $|z = 0\rangle$ with the Hamlitonian B for a time b. B is the adjacency matrix of a big graph whose vertices correspond to the independent sets of the input graph and whose edges can be read off (39). We view this as a continuous time quantum walk entering the big graph at a particular vertex [5]. In the extreme case where the input graph has no edges, all strings of length n represent independent sets so the Hilbert space dimension is 2^n. In this case B is the adjacency matrix of the hypercube, realizable as in (3). Setting $b = \pi/2$, the state (41) (with $p = 1$ there is only one unitary) is $|z = 11\ldots11\rangle$ which maximizes the objective function. In the more general case we can view (41) as a succession of quantum walks punctuated by applications of C dependent unitaries which aid the walk in achieving its objective. The algorithm of the previous sections can also be viewed this way although the starting state is not a single vertex.

VIII. CONCLUSION

We introduced a quantum algorithm for approximate combinatorial optimization that depends on an integer parameter p. The input is an n bit instance with an objective function C that is the sum of m local terms. The goal is to find a string z for which $C(z)$ is close to C's global maximum. In the basic algorithm, each call to the quantum computer uses a set of $2p$ angles $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ and produces the state

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = U(B, \beta_p)\, U(C, \gamma_p) \cdots U(B, \beta_1)\, U(C, \gamma_1)\ |s\rangle\,. \tag{48}$$

This is followed by a measurement in the computational basis yielding a string z with an associated value $C(z)$. Repeated calls to the quantum computer will yield a good estimate

of

$$F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \boldsymbol{\gamma}, \boldsymbol{\beta}| \, C \, |\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle \, . \tag{49}$$

Running the algorithm requires a strategy for a picking a sequence of sets of angles with the goal of making F_p as big as possible. We give several possible strategies for finding a good set of angles.

In section II we focused on fixed p and the case where each bit is in no more than a fixed number of clauses. In this case there is an efficient classical algorithm that determines the best set of angles which is then fed to the quantum computer. Here the quantum computer is run with only the best set of angles. Note that the "efficient" classical algorithm which evaluates (25) using (24) could require space doubly exponential in p.

An alternative to using a classical preprocessor to find the best angles is to make repeated calls to the quantum computer with different sets of angles. One strategy, when p does not grow with n is to put a fine grid on the compact set $[0, 2\pi]^p \times [0, \pi]^p$ where the number of points is only polynomial in n and m. This works because the function F_p does not have peaks that are so narrow that they are not seen by the grid.

The QAOA can be run on a quantum computer with p growing with n as long as there is a strategy for choosing sets of angles. Perhaps for some combinatorial optimization problem, good angles can be discovered in advance. Or the quantum computer can be called to evaluate $F_p(\boldsymbol{\gamma}, \boldsymbol{\beta})$, the expectation of C in the state $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$. This call can be used as a subroutine by a classical algorithm that seeks the maximum of the smooth function $F_p(\boldsymbol{\gamma}, \boldsymbol{\beta})$. We hope that either p fixed or growing slowly with n will be enough to have this quantum algorithm be of use in finding solutions to combinatorial search problems beyond what classical algorithms can achieve.

IX. ACKNOWLEDGEMENTS

This work was supported by the US Army Research Laboratory's Army Research Office through grant number W911NF-12-1-0486, and the National Science Foundation through grant number CCF-121-8176. The authors thank Elizabeth Crosson for discussion and help in preparing the manuscript. We also thank Cedric Lin and Han-Hsuan Lin for their help. EF would like to thank the Google Quantum Artificial Intelligence Lab for discussion and

support.

---

[1] Eran Halperin, Dror Livnat, Uri Zwick.
MAX CUT in cubic graphs, 2004.
Journal of Algorithms, Volume 53 Issue 2, Pages 169-185.

[2] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Michael Sipser.
Quantum computation by adiabatic evolution, 2000.
arXiv:quant-ph/0001106.

[3] Edward Farhi, Jeffrey Goldstone, Sam Gutmann.
Quantum Adiabatic Evolution Algorithms versus Simulated Annealing, 2002.
arXiv:quant-ph/0201031.

[4] Elizabeth Crosson, Edward Farhi, Cedric Yen-Yu Lin, Han-Hsuan Lin, Peter Shor.
Different strategies for optimization with the quantum adiabatic algorithm, 2014.
arXiv:1401.7320 [quant-ph].

[5] Edward Farhi, Sam Gutmann.
Quantum Computation and Decision Trees, 1997.
Phys. Rev. A 58, 915 arXiv:quant-ph/9706062.