Quantum Approximate Optimization with Hard and Soft Constraints

Stuart Hadfield[*], Zhihui Wang[+,**], Eleanor G. Rieffel[+],

Bryan O'Gorman[+,†], Davide Venturelli[+,**], Rupak Biswas[+]

[*] Department of Computer Science, Columbia University, New York, NY

[+] Quantum Artificial Intelligence Lab., NASA Ames Research Center, Moffett Field, CA

[**] Universities Space Research Association, Mountain View, CA

[†]Stinger Ghaffarian Technologies, Inc., Greenbelt, MD

ABSTRACT

Challenging computational problems arising in the practical world are frequently tackled by heuristic algorithms. Small universal quantum computers will emerge in the next year or two, enabling a substantial broadening of the types of quantum heuristics that can be investigated beyond quantum annealing. The immediate question is "What experiments should we prioritize that will give us insight into quantum heuristics?" One leading candidate is the quantum approximate optimization algorithm (QAOA) metaheuristic. Here, we provide a framework for designing QAOA circuits for a variety of combinatorial optimization problems with both hard constraints that must be met and soft constraints whose violation we wish to minimize. We work through a number of examples, and discuss design principles and implementation considerations.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

KEYWORDS

quantum computing, optimization algorithms

ACM Reference format:

Stuart Hadfield[*], Zhihui Wang[+,**], Eleanor G. Rieffel[+], Bryan O'Gorman[+,†], Davide Venturelli[+,**], Rupak Biswas[+]. 1997. Quantum Approximate Optimization with Hard and Soft Constraints. In *Proceedings of Post Moore's Era Supercomputing, Denver, Colorado, USA, November 2017 (PMES'17),* 7 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Over the last few decades, researchers have discovered several stunning instances [18] of quantum algorithms that provably outperform the best existing classical algorithms and, in some cases, the best possible classical algorithm. For most problems, however, it is currently unknown whether quantum computing can provide an advantage, and if so, how to design quantum algorithms that realize such advantages. Today, challenging computational problems arising in the practical world are frequently tackled by heuristic algorithms, which by definition have not been analytically proven to be the best approach, or even proven to outperform the best approach of the previous year. Rather, these algorithms are empirically shown to be effective, by running them on characteristic sets of problems. Only now that prototype quantum hardware is becoming available is this approach to algorithm design open for quantum algorithms.

Previously, the only quantum hardware available was special-purpose quantum hardware, specifically quantum annealers targeting combinatorial optimization problems. The emerging gate-model processors, currently in prototype phase, are universal in that, once scaled up, they can run any quantum algorithm. Within the last year, IBM has made available publicly through the cloud a 5-qubit gate-model chip [11], and announced recently an upgrade to a 17-qubit chip. Other groups, such as Google [3], Rigetti Computing [20], TU Delft, and UC Berkeley,with Google and Rigetti anticipate providing processors with 40-100 qubits within a year or two.[16] Gate-model computing expands the potential applications beyond optimization, as well as enabling a broader array of quantum approaches to optimization.

While limited exploration of quantum heuristics beyond quantum annealing (QA) has been possible through small-scale classical simulation, the exponential overhead in such simulations has limited their usefulness. The next decade will see a blossoming of quantum heuristics as a broader and more flexible array of quantum computational hardware comes into being. The immediate question is "What experiments should we prioritize that will give us insight into quantum heuristics?" One leading candidate is QAOA circuits.

QAOA circuits were first proposed by Farhi *et al.* [6] as the basis for a quantum approximate optimization algorithm (QAOA), and a number of tantalizing results have been obtained since [12, 21–23]. QAOA circuits have a particularly simple form, alternating between cost-function based operations and mixing operations. Prior work focused almost exclusively on cases in which there are no hard constraints and the mixing term takes an exceptionally simple form, though the initial paper [6] included a section on a variant of the algorithm that provides an example that suggests how the algorithm can be generalized to more complex situations, particularly ones in which not all bit strings are feasible solutions.

Here, we further develop generalizations of the algorithm to combinatorial optimization problems with both hard constraints, which must be satisfied, and soft constraints, to which we want to maximize compliance. One appealing aspect of the QAOA approach

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
PMES'17, November 2017, Denver, Colorado, USA
© 2017 Copyright held by the owner/author(s).
ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

is that, like quantum annealing, it is relatively easy for a computer scientist with little knowledge of quantum computing to design a QAOA circuit for a given approximate optimization problem that could be run on near-term hardware. After reviewing the QAOA algorithm, we describe a framework for designing cost functions incorporating the soft constraints and mixing operators enforcing the hard constraints. The design criteria aim to provide efficient schemes by using mixing terms that are restricted to the feasible subspace and thus do not spend time exploring infeasible solutions. We provide analytical and numerical results that support the choice to use the mixing operations to enforce the hard constraints rather than incorporating them into the cost function. We then provide detailed examples, focusing on various optimization versions of scheduling problems, followed by a brief discussion of other problems. We conclude with suggestions for next steps, including the potential for realizing some of these algorithms on near-term hardware, with an eye to estimating their ultimate impact.

2 BACKGROUND ON QAOA

QAOA circuits iteratively alternate between a step dependent on the optimization cost function and a mixing term. Level p QAOA circuits, $QAOA_p$, iterate p times. We will refer to circuits with the above structure as QAOA circuits whether they are used for approximate optimization or for some other purpose. We define QAOA circuits formally in Sec. 3.2. Here, we motivate this work by giving an an overview of prior results on QAOA.

Farhi *et al.* [6] proposed the quantum approximate optimization algorithm (QAOA) as a metaheuristic with the potential to improve on classical approximation algorithms by providing better approximation ratios or by finding solutions achieving those ratios more efficiently than classical algorithms. Indeed, Farhi *et al.* [7] were able to obtain bounds for a simple QAOA algorithm that beat the best approximation ratio for all known efficient classical algorithms for the problem E3Lin2, only to inspire a better classical algorithm [1] that narrowly beats the approximation ratio for the QAOA_1 algorithm by a log factor. More advanced QAOA algorithms, with more repetitions, could potentially beat the approximation ratio of this classical algorithm, but such an analysis is challenging.

Since Farhi *et al.*'s original work, QAOA circuits have also been applied for exact optimization [12, 23] and sampling [8]. Wecker *et al.* [23] explores learning parameters for QAOA circuits on instances of MAX-2-SAT that result in high overlap with the optimal state. Jiang *et al.* [12] demonstrates that the class of QAOA circuits is powerful enough to obtain the $\Theta(\sqrt{2^n})$ query complexity on Grover's problem, and also provides the first algorithm within the QAOA framework to show a quantum advantage for a number of iterations p in the intermediate range between $p = 1$ and $p \to \infty$. Farhi and Harrow [8] proved that, under reasonable complexity theoretic assumptions, it is not possible for any classical algorithm to produce samples according to the output distribution of QAOA circuits, with those with just a single iteration ($p = 1$). Their results suggest that QAOA circuits applied to sampling are among the most promising candidates for early demonstrations of "quantum supremacy" [2, 17]. It remains an open question whether QAOA circuits provide a quantum advantage for approximate optimization.

Because these circuits have uses beyond approximate optimization, and people tend to refer to the QAOA algorithm, even though the final A stands for "algorithm," we propose that the acronym be reworked so as to describe the structure of these circuits: the Quantum Alternating Operator with adjusting Angles (QAOA) circuits.

3 QAOA FRAMEWORK

Combinatorial optimization problems are often represented in a form in which, in addition to a cost function to be maximized, there are *hard constraints* which must be satisfied in order for the solution to be valid. We consider only representations in which the variables are binary; the cost function is a map $C : \{0,1\}^n \to \mathbf{R}$. Let n be the number of variables. We use *search space* to refer to the full set of n-bit strings. Any bit string that satisfies all hard constraints is called a *feasible solution*, and the set of such bit strings is the *feasible subset* of the search space. In an *exact optimization* problem, the goal is to find a bit string that maximizes the cost function. In an *r-approximate optimization* problem, the goal is to find a bit string $\mathbf{x}$, such that $C(\mathbf{x})$ is within a factor of r of the maximum:

$$\frac{C(\mathbf{x})}{C_{max}} \geq r. \tag{1}$$

An algorithm is an r-approximation algorithm for problem, if for every instance of the problem, the algorithm finds a bit string with cost function within a factor of r of the maximum.

The most explored general-purpose quantum algorithms for optimization, adiabatic quantum optimization (AQO),quantum annealing,and the quantum approximate optimization algorithm (QAOA), are expressed in terms of two Hamiltonians applied to an n-qubit quantum register. We briefly review key concepts in quantum computing that we will use in this paper. Please see a standard textbook, or review article, such as [18], for an introduction to this topic.

3.1 A brief review of quantum computing

A qubit takes values in a 2-dimensional complex vector space, with vectors that are equivalent up to scaling by a complex number considered the same state. (The zero vector is not a legitimate qubit value.) In quantum mechanics, Dirac's bra/ket notation is used, where $|v\rangle$ is notation for a column vector, and $\langle v|$, its Hermitian conjugate, a row vector. Two orthogonal vectors, the *computational basis states* $|0\rangle$ and $|1\rangle$ of the qubit, are chosen to represent classical bit values. We use the convention $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Multiple qubit spaces combine via the tensor product. States of multiple qubits are non-zero vectors in this tensor product space, up to scaling by a complex number. One basis for the states in a n qubit system, the *computational basis*, is made up of the states that are tensor products of the single-qubit computational basis states, $|b_{n-1}b_{n-2}\dots b_0\rangle = |b_{n-1}\rangle \otimes \cdots \otimes |b_0\rangle$, where $b_i \in \{0,1\}$. For convenience we will also use the label $|x\rangle$, for x a non-negative integer, to refer to the state vector labeled with the binary string corresponding to x. All n-qubit states can be written as a *superposition* of such states computational basis states. $\sum_{x=0}^{N-1} a_x|x\rangle$, where $N = 2^n$ is the number of n-bit strings, and the a_x are referred to as *amplitudes*. Each amplitude is a complex number $a_x = |a_x|e^{i\theta}$, with real *magnitude* $|a_x|$ and *phase* $e^{i\theta}$.

Operations on n-qubit registers are represented as unitary transformations on the 2^n-dimensional state space. Any unitary transformation U can be written as $U = e^{iH}$ for some Hermitian operator H called the Hamiltonian. Some useful single qubit operators include the Pauli operators $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$, and $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, which together with the identity transformation I, form a basis for linear operators on a 2-dimensional complex vector space. The X operator corresponds the the classical NOT operator, while the other two operators are quantum operators that affect the relative phase of the $|0\rangle$ and $|1\rangle$ components of a qubit's state, which can be useful for setting up quantum interference effects in the subsequent computation. We will make use of the Pauli communtation relations $[X, Y] = -[Y, X]$, $[Y, Z] = -[Z, Y]$, and $[Z, X] = -[X, Z]$, where $[A, B] = AB - BA$ is the commutator. Since the Pauli operators are both Hermitian and unitary, they can be used as Hamiltonians or as unitary operators directly. Tensor products of Pauli operators act as unitary operators on a multiqubit system. Because the eigenvectors of Z are $|0\rangle$ and $|1\rangle$, Hamiltonians made up of of Z operators are considered "classical" Hamiltonians. For example, Z_1X_3 applies Z to the first qubit and X to the third qubit. Sums of Pauli operators are not unitary, in general, but are Hermitian so may be used as Hamiltonians for unitary operators. A useful property of Hamiltonians H that are both unitary and Hermitian is that $e^{i\theta H} = \cos(\theta)I + i \sin\theta H$, where for $\theta = \pi/2$, we have H up to an irrelevant global phase of i.

3.2 A brief review of QAOA circuits

Let $C : \{0, 1\}^n \rightarrow \mathbb{R}$ be a cost function. To represent the cost function as a problem Hamiltonian H_C, we follow the standard practice of substituting $\frac{1}{2}(I + Z)$ for each binary variable. Since such a Hamiltonian contains only Z terms, problem Hamiltonians are "classical" Hamiltonians, and the computational basis vectors are eigenvectors of H_C. Since $|0\rangle$ is a 1-eigenvector for Z, and $|1\rangle$ is a -1-eigenvalue for Z, the eigenvalue of H_C for eigenvector $|x\rangle$ is the value of the cost function on the bitstring $\mathbf{x}$, $C(\mathbf{x})$. We will refer to the subspace spanned by $|\mathbf{x}\rangle$, where $\mathbf{x}$ lies in the feasible subset, as the *feasible subspace*.

A $QAOA_p$ circuit loops p times, and on iteration i applies two Hamiltonians in turn,

- a cost-function-based Hamiltonian H_C, for time β_i, and
- mixing Hamiltonian H_M, for time γ_i.

The first achieves *phase separation*, with computational basis states $|x\rangle$ receiving a phase depending on the associated cost $C(\mathbf{x})$. The second *mixes amplitude* between computation basis states. When used for optimization, the idea is to choose β_i and γ_i such that quantum interference results in a concentration of amplitude in computational basis states with high cost. At the end of the algorithm, a quantum measurement is done in the computational basis, resulting in a single bit string as output, depending probabilistically on amplitudes in the final quantum state. Specifically, $\mathbf{x}$ is obtained with probability $|a_x|^2$, the square of its magnitude in the final state. As we discuss below, the choice of initial state is also important.

3.3 Framework for mapping combinatorial optimization problems to QAOA

All problem information must be encapsulated in some way in the Hamiltonians. For the hard constraints, we have a choice between incorporating them in the cost Hamiltonian or the mixing Hamiltonian. Because current quantum annealers have a fixed driver (the mixing Hamiltonian in the quantum annealing (QA) setting), all problem dependence must be captured in the cost Hamiltonian. The general strategy is to incorporate the hard constraints as penalty terms in the cost function, and then convert the cost function to a cost Hamiltonian. This approach generates mappings for an optimization problem that works with any mixing Hamiltonian, hence a uniform mixing Hamiltonian can be implemented (as is on current quantum annealers) universally for all problems. It has a number of disadvantages even in that setting,one major one being that search takes place over the full search space rather than being restricted to the feasible subset.

In this paper we take a different approach, incorporating the hard constraints into the mixing Hamiltonian so as to constrain the search to the feasible subspace. Specific examples will be discussed in Sec. 4. Here, we provide a general framework, taking inspiration from the example developed in Sec. VII of Farhi *et al.* [6], and build on a theory developed by Hen & Spedalieri [10] and Hen & Sarandy [9] for use in the AQO setting. Because in AQO the ground subspace in preferred, most problems are phrased as minimization problems, whereas in the QAOA setting, most problems are phrased in terms of maximization problems, since the initial examples were all cases in which the cost function was simply a sum of constraints to be satisfied. As a result, the cost functions, and resulting Hamiltonians, differ by a sign between the two settings.

The mixing Hamiltonian should both restrict exploration to the feasible subspace and promote exploration of that subspace. Following [9], a mixing Hamiltonian must

- preserve the feasible subspace, so that the resulting unitary takes states in the feasible subspace to states in the feasible subspace, and
- provide a mixing (or hopping) mechanism, that enables full exploration of the feasible subspace from any starting state in the feasible subspace.

In the QAOA circuit setting, some of the design criteria for AQO (not listed) in [10] are not needed, enabling us to have greater flexibility in the design of the mixing Hamiltonian. [1]

For the design of the problem and mixing Hamiltonians, it is sometimes useful to define a Hamiltonian encapsulating the hard constraints, H_A, with the feasible subspace as its ground subspace. Any Hamiltonian that commutes with H_A will preserve the feasible subspace. A mixing Hamiltonian should not commute with the cost function Hamiltonian H_C, since it must mix between states with different eigenvalues to move amplitude into subspaces of the feasible space corresponding to high values of C. Thus, as design criteria for the mixing Hamiltonian H_M, we have $[H_M, H_A] = 0$ and $[H_M, H_C] \neq 0$. Further, we need to ensure that H_M contains mechanisms for exploring the entire space.

[1] Because high weight operators translate to unitaries that are implementable in the circuit model setting, we do not have to restrict ourselves to low weight Hamiltonians. Nor does it need to be gapped.

In Sec. 4, we introduce mixing Hamiltonians for a variety of problems. Here, we describe one tool that will be useful, in various forms, for constructing such Hamiltonians. Hen *et al.* [10] explore, in the AQO setting, terms of the form $SWAP_{ij} = \frac{1}{2}(I + X_iX_j + Y_iY_j + Z_iZ_j)$. When applied to two-qubit computational basis states on qubits i and j, it takes $|0_i1_j\rangle$ to $|1_i0_j\rangle$ and vice versa, while leaving $|0_i0_j\rangle$ and $|1_i1_j\rangle$ unchanged. Thus, the operation swaps the bit values of the two qubits in the computational basis, generalizing the classical SWAP operation. Since $SWAP_{ij}$ is Hermitian as well as unitary, it can also be used as a Hamiltonian, and with $e^{i\theta SWAP_{ij}} = \cos(\theta)I + i\sin\theta SWAP_{ij}$, which applies the $SWAP_{ij}$ to the two qubits for $\theta = \pi/2$, and is a combination of the identity and the $SWAP_{ij}$ for other values of θ. The SWAP operation preserves the Hamming weight of bit strings. We will see that it will be used to incorporate various hard constraints, such as exactly one of a set of binary variables can take value 1, into mixing Hamiltonians. Because, the I and Z_iZ_j terms commute with any classical Hamiltonian, and thus with all cost function Hamiltonians H_C, we will often just use terms of the form $\frac{1}{2}(X_iX_j + Y_iY_j)$, which happens to be relatively easy to implement on superconducting quantum hardware.

4 MAPPING GRAPH COLORING OPTIMIZATION PROBLEMS TO THE QAOA FORMALISM

Graph-k-Coloring is an important NP-complete (for $k \geq 3$) problem with many applications, such as scheduling [14, 19] and memory allocation [4] problems. Given a (undirected) graph $G = (V, E)$, Graph-k−Coloring asks whether there exists an assignment of one of k colors to each vertex such that every edge is properly colored (connects two vertices of different color). If such an assignment exists, the graphs is said to be *k-colorable.*

Several optimization variants of Graph-k-Coloring are known:

- **Maximize properly colored edges.** Given a graph, find a color assignment such that the number of properly colored edges is maximized.
- **Maximal Properly-Colorable Induced Subgraph.** Given a graph, determine the maximum number of vertices for which all edges in the graph can be properly colored.
- **Chromatic number.** Given a graph, determine its chromatic number, the minimum number of colors required to properly color the graph.

The first is a generalization of MaxCut, the second of Independent Set. As we will see, there exists a mixing Hamiltonian for the first that has a relatively simple form, and is independent of the problem instance (other than size). Mixing Hamiltonians for the second must be more complicated, and while their general form is problem instance dependent, each problem instance has a specific mixing Hamiltonian.

4.1 Maximizing the properly colored edges

Problem: Given a graph $G = (V, E)$ with n vertices and m edges, we seek a k-color assignment that maximizes the number of properly colored edges.

There are many ways to represent this problem on a quantum computer, with various trade-offs. We employ a reasonable compromise, a unary representation which uses k qubits per vertex, as has been used in quantum annealing [9, 10, 15, 19]. Color assignments are encoded using kn binary variable $x_{v,i}$, with $x_{v,i} = 1$ indicating that vertex v has been assigned color i. For each vertex, a hard constraint is that the vertex be assigned exactly one color. Feasible bit strings, satisfying these n constraints, correspond to kn-bit strings in which, for each vertex v, exactly one of its k variables $x_{v,i}$ is set to one. The cost function is

$$m - \sum_{<uv> \in E} \sum_{i=1}^{k} x_{u,i}x_{v,i}, \tag{2}$$

which penalizes every improperly colored edge. Substituting $\frac{1}{2}(I + Z)$ for each binary variable, we obtain

$$H_C = mI - \frac{km}{4}I + \frac{1}{4}\sum_{<uv>}\sum_{i=1}^{k}(Z_{u,i}Z_{v,i} - Z_{u,i} - Z_{v,i}). \tag{3}$$

Feasible bit strings satisfy the n constraints $\sum_{i=1}^{k} x_{v,i} = 0$, one for each vertex v. We rewrite them into

$$(1 - \sum_{i=1}^{k} x_{v,i})^2 = 0,$$

so that the translated Hamiltonian

$$H_A = -\frac{1}{2}\sum_{v}\left((k-2)\sum_{i=1}^{k} Z_{v,i} - \sum_{i\neq j}^{k} Z_{v,i}Z_{v,j}\right) \tag{4}$$

admits the feasible subspace as its ground subspace.

We seek a mixing Hamiltonian H_M, that meets the criteria laid out in Sec. 3.3, keeping the evolution within the feasible subspace. For each vertex v, we include a term of the form

$$B_v = \frac{1}{k}\sum_{i=1}^{k} X_{v,i}X_{v,i+1} + Y_{v,i}Y_{v,i+1}, \tag{5}$$

known in physics as the XY Model on a ring. The entire mixing Hamiltonian is $H_M = \sum_v B_v$. From the Pauli commutation relations, we have $[H_M, H_A] = 0$ and $[H_M, H_C] \neq 0$ as desired. Using the intuition from Sec.3.3, each term generates transitions between colors on vertex v through its connection with the SWAP operator, and thus, starting from any feasible state, all feasible states are reachable after sufficient repetitions p.

We have a choice in starting state. While any feasible state will do, we expect the algorithm to be more efficient if we start with a superposition of all feasible states, in which a vertex is assigned all colors with equally probablility. This hypothesis will be tested in future research, and tradeoffs between ease of implementation of a starting state and its effectiveness as a starting point will be evaluated.

We could also have used a mixing term in which B_v contains more connections, so that the colors for each vertex are not just connected to adjacent colors in the indexing, but to more colors as well. This provides more channels for colors to propagate, we hence expect it to lead to better performance of the algorithm. But more connections require more couplings between qubits, and poses a higher resource requirement to implement on the hardware. For example, with a fully connected graph for each vertex, the number of connections per qubit scales with the number of colors instead of being constant as is the case for the ring.

Directly extending the MaxCut application of [6], the color of each vertex could be encoded with $\lceil \log_2 k \rceil$ qubits. While such an encoding efficiently represents the required states, the problem Hamiltonian would be much more complicated, requiring many way couplings to penalize adjacent vertices colored with the same color. Since usually $k \ll n$, our approach does not add unreasonable overhead to the problem representation, and we expect it to be significantly easier to implement in practice.

4.2 Finding a Maximal Properly-Colorable Induced Subgraph

The *induced subgraph* of a graph $G = (V, E)$ for a subset of vertices $W \subset V$ is the graph $H = (W, E_W)$, where the edge between vertices w_i and w_j in W is included in E_W if and only if that edge is in E.

Problem: Given a graph $G = (V, E)$ with n vertices and m edges, find the largest induced subgraph that can be properly k-colored.

We represent colorings as in Sec.4.1, with variables $x_{v1}, \ldots, x_{vk}$, but with one additional variable x_{v0} per vertex to represent an "uncolored" vertex, indicating that the vertex will not be part of the induced subgraph.

In this case, feasible strings, in addition to having each vertex uniquely colored (or assigned as uncolored), must correspond to proper colorings on the induced graph on the colored vertices. Thus, the mixing term will be more complicated than for the previous problem, essentially incorporating information that was in the cost function. The cost function now takes a simple form:

$$C = \sum_v \sum_{i=1}^{k} x_{vi}, \tag{6}$$

the number of colored nodes. The variables x_{v0} are not included in the sum since they correspond to uncolored nodes that are not part of the induced subgraph. The corresponding Hamiltonian is

$$C = \frac{1}{2} mI - \frac{1}{2} \sum_u \sum_{i=1}^{k} Z_{ui}. \tag{7}$$

To design the mixing term, consider the transitions between feasible states. A given vertex can be feasibly colored i only if none of its adjacent vertices are also colored i. Thus, the transition rule at each vertex must depend on the local graph topology and colorings. Consider the controlled operation

$$(\bar{x}_{v_1 i} \bar{x}_{v_2 i} \ldots \bar{x}_{v_\ell i}) \cdot SWAP(x_{u0}, x_{ui}), \tag{8}$$

where $v_1, \ldots, v_\ell$ are neighbors of vertex u in graph G. This operation changes the color of vertex u from uncolored to colored with color i (or vice versa), if and only if none of its neighbors are colored with color i. The corresponding Hamiltonian term (after dropping the terms for the SWAP that have no effect), is

$$B_{ui} = \frac{1}{2^{\ell+1}} (X_{u0} X_{u1} + Y_{u0} Y_{u1}) \prod_{j=1}^{\ell} (I + Z_{v_j i}). \tag{9}$$

The overall mixing Hamiltonian is $B = \sum_u \sum_i B_{ui}$. Since B contains the means to color a vertex with color i if none of its neighbors are colored with color i, and a means to uncolor a vertex (as long as none of its neighbors share its current color, which is always the case in the feasible subspace), the mixing term enables exploration of the full feasible subspace starting from any state in that subspace.

4.3 Finding a Graph's Chromatic Number

The chromatic number of a graph $G = (V, E)$ is the minumum number k^* of colors that can properly color a graph. **Problem:** Given a graph $G = (V, E)$, minimize the number of colors to properly color it.

Any n-vertex graph may be trivially n-colored. Thus, without loss of generality assume we know a quantity ℓ, with $k^* \leq \ell \leq n$, such that an explicit ℓ-coloring of G is known (found e.g. by some classical algorithm). We then represent our problem using ℓ qubits per vertex, with the Hamming weight 1 subspace for each vertex encoding its possible color assignments.

We define the feasible states to be those encoding proper ℓ-colorings, many of which will use less than ℓ colors. Assume we have restricted to the feasible subspace.

For the mixing term, we use a controlled operation similar to Eq. (8) in Sec. 4.2 only now supporting re-coloring a vertex v with color i when no neighboring vertices are assigned color i.

For a given state, the function $\prod_{u \in V} \bar{x}_{u,j}$ gives 1 only if no vertex is colored j. Thus, we seek to maximize the number of unused colors, encoded by the n-local objective Hamiltonian

$$C = \sum_{j=1}^{\ell} \prod_{u \in V} \bar{x}_{u,j} = \frac{1}{2^n} \sum_{j=1}^{\ell} \prod_{u \in V} (I + Z_{u,j}).$$

4.4 Traveling Salesman Problem (TSP)

A *vertex tour* of a complete graph $G = (V, E)$ is a subset $E' \subset E$ such that every vertex in $G' = (V, E')$ has degree 2 (i.e. gives a route for the salesman to visit each vertex exactly once). **Problem:** Given a complete graph $G = (V, E)$ and distances $d_{uv} \in \mathbb{R}$, find the minimial tour length.

We represent vertex tours with n^2 binary variables $\{x_{vj}\}$ indicating whether vertex v is visited at the jth stop of the tour, $j = 1, \ldots, n$. Feasible strings are those representing valid tours, i.e. for each v have $\sum_{j=1}^{n} x_{vj} = 1$ (each v visited exactly once), and for each position j have $\sum_{v \in V} x_{vj} = 1$ (a single vertex visited at each stop). The objective Hamiltonian gives the tour length and on feasible states reduces to

$$C = \ell I + \sum_{(uv) \in E} \sum_{j=1}^{n} d_{uv} (Z_{uj} Z_{v(j+1)} + Z_{u(j+1)} Z_{vj}),$$

where $\ell = (2 - n/2) \sum_{(uv) \in E} d_{uv}$.

As feasible states can be seen as $n \times n$ matrices with a single 1 in every column or row, a mixing Hamiltonian may be constructed as a sum of row swaps $B = \sum_{u<v} B_{uv}$,

$$B_{uv} = \prod_{j=1}^{n} (X_{uj} X_{vj} + Y_{uj} Y_{vj})$$

which clearly preserves feasibility. Alternatively, we can reduce to 4-local terms

$$B' = \sum_{i\neq j}\sum_{u\neq v} |0_{ui}1_{uj}1_{vi}0_{vj}\rangle\langle 1_{ui}0_{uj}0_{vi}1_{vj}| + h.c. \quad (10)$$

$$= \sum_{i\neq j}\sum_{u\neq v} S^-_{u,i}S^-_{v,j}S^+_{u,j}S^+_{v,i} + h.c.\,. \quad (11)$$

where in the second line we introduced the creation and annihilation operators $S^+ = X + iY = |1\rangle\langle 0|$ and $S^- = X - iY = |0\rangle\langle 1|$.

4.5 Single Machine Scheduling (SMS)

We consider the SMS problem to minimize the total tardiness, $(1||\sum T_j)$, which is NP-hard.[5, 13] **Problem:** Given n jobs, each with a running time p_j and deadline d_j, to run on a single machine that takes one job at a time, find a schedule that minimizes the total tardiness $T = \sum_{j=1}^n T_j$, where $T_j \geq 0$ gives the amount of time job j is late. All parameters are taken to be integers.

For each job j, we use a binary variable $x_{j,t}$ to denote whether job j starts at time t, where $t = 0, \ldots, T$, and $T = \sum_j p_j - \min_j\{p_j\}$. Feasible strings are those for which each job is assigned a single start time, i.e. $\sum_t x_{j,t} = 1$. The cost function, i.e., the total tardiness becomes

$$C = \sum_j \sum_{t>(d_j-p_j)} x_{j,t}(t + p_j - d_j), \quad (12)$$

where the sum over t is taken over only times later than the latest time that the job can start without being late, $d_j - p_j$. As a result, whenever $x_{j,t} = 1$ for such a t, the lateness of job j is added into the sum. Any permutation of $(1, 2, \cdots, n)$ as the order of jobs maps to a feasible initial state.

We now design a mixing Hamiltonian in the flavor of the "bubble sort": we allow the swapping of the order of any two neighboring jobs. This guarantees a path to any ordering, hence the accessibility to the whole feasible subspace. Consider job j starting at t and the next job j' would start at $t + p_j$, swapping these two jobs would lead to job j' starting at time t and job j starting at $t + p_{j'}$. This can be realized by the Hamiltonian term $S^-_{j,t}S^-_{j',t+p_j}S^+_{j,t+p_{j'}}S^+_{j',t}$. Therefore the mixing Hamiltonian is

$$H_B = \sum_{j\neq j'}\sum_{t=0}^{T} S^-_{j,t}S^-_{j',t+p_j}S^+_{j,t+p_{j'}}S^+_{j',t} + h.c.\,. \quad (13)$$

5 EMPIRICAL CONFIRMATION FOR MIXING TERMS ENFORCING HARD CONSTRAINTS

To further support our analytical reasons for prefering mixing terms enforcing hard constraints, we explored different approaches to QAOA on a simple problem, the ring of disagrees, which can be viewed as 2-coloring on a circular graph with n vertices. For this problem, numerical results for small p were obtained in [6], with Wang *et al.* [22] advancing both the numerical results and the analytical understanding of the problem through a fermionic point of view.

A binary encoding with one binary indicating whether the vertex is colored one way or the other is natural in this case. Using the unary encoding as we did in Sec. 4.1 introduces "one vertex is colored exactly one color" as hard constraints, and thus supports a study of different ways of handling those constraints in a simple setting.

We compared three mappings

(1) Binary encoding and $\sum_i X_i$ as mixing Hamiltonian
(2) Mapping described in .4.1: unary encoding, Eq. (2) as the cost function and Eq. (5) as the mixing term;
(3) Unary encoding, a cost function incorporating penalty terms, $C = \frac{m}{2}I + \frac{1}{4}\sum_{i=1}^n \sum_{c\neq c'}^1 (Z_{i,c} + Z_{i+1,c'} - Z_{i,c}Z_{i+1,c'}) + c_{pen}\sum_{i=1}^n \sum_c Z_{i,c}Z_{i+1,}$ and the simple mixing Hamiltonian $\sum_i X_i$.

(The vertices in the ring are numbered $1, \ldots, n$ and we use modulo n arithmetic on the indices.)

We confirmed analytically for $p = 1$ and numerically for small p that the same approximation ratio can be obtained using Mapping-(1) and Mapping-(2), while Mapping-(3) yields much poorer results.

For Mapping-(3), we consider two initial states, a simple form $|s\rangle = |+\rangle^{\otimes n}$ and $|s'\rangle = (|01> + |10\rangle)^{\otimes n}$ in the feasible subspace.

For QAOA$_1$, starting with the initial state $|s'\rangle$ in the feasible subspace, one gets no improvement than the initial state. The expectation value for the approximation ratio of the initial state is $1/2$ (equivalent to randomly guessing a feasible bit string), and remains $1/2$ for QAOA$_1$ with any parameter setting. For higher p, the algorithm can give better approximation ratios than $1/2$, but for the cases we tried, the ratio was worse than for the approach III.

For the initial state $|s\rangle = |+\rangle^{\otimes n}$, simply measuring the initial state results in states which do not correspond to valid colorings (some vertices are colored with two colors or with none at all), so that the expectation value is much less than $1/2$ (but depends on the penalty weigh in the cost function), and while a $QAOA_1$ algorithm with the right parameters improves the expectation value, it is still worse that randomly guessing a feasible bit string. The ratio gets worse with size since the feasible subspace grows as 2^n, but the whole Hilbert space as 4^n, where n is the number of vertices. So the approach starting in the feasible superposition $|s'\rangle$ performs better than the one starting in the standard initial state $|s\rangle$, though both substantially underperform the approach we recommend here, that of incorporating the hard constraints into the mixing operator.

6 CONCLUSIONS

We are preparing a longer version of this paper, detailing mappings for other problems with both hard and soft constraints. Some are straightforward generalizations of the mappings given above. For example, constructions for graph coloring with weights, and graph coloring with different color sets, are easy generalizations of the construction we give above for maximizing the number of propertly colored edges, and make use of the XY Hamiltonian. Constructions for graph partitioning, maximum vertex k-cover, and maximal bisection. As mentioned above, finding the maximally colorable induced subgraph is equivalent to max independent set. This construction easily generalizes to other problems such as max clique and minimum vertex cover, which make use of the controlled-SWAP Hamiltonian.

Future directions include:

- Investigating the performance of QAOA under drivers with more or less connectivity with the aim of understanding tradeoff in terms of algorithmic efficiency and ease of implementability.
- Investigating how different initial states contribute to the efficiency of the algorithm.
- Development of parameter setting strategies, and investigating how different formulations of the same problem affect the ease with which good parameters can be found.

7 ACKNOWLEDGEMENTS

The authors would like to acknowledge support from the NASA Advanced Exploration Systems program and the NASA Ames Research Center. S.H. was supported by NASA under award NNX12AK33A. He would like to thank the Universities Space Research Association (USRA), as well as the NASA Ames Research Center, for the opportunity to participate in the program that enabled this research.

REFERENCES

[1] Boaz Barak, Ankur Moitra, Ryan O'Donnell, Prasad Raghavendra, Oded Regev, David Steurer, Luca Trevisan, Aravindan Vijayaraghavan, David Witmer, and John Wright. 2015. Beating the random assignment on constraint satisfaction problems of bounded degree. *arXiv:1505.03424* (2015).

[2] Sergio Boixo, Sergei V. Isakov, Vadim N. Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, John M. Martinis, and Hartmut Neven. 2016. Characterizing Quantum Supremacy in Near-Term Devices. *arXiv:1608.00263* (July 2016). http://arxiv.org/abs/1608.00263

[3] Sergio Boixo, Vadim N. Smelyanskiy, Alireza Shabani, Sergei V. Isakov, Mark Dykman, Vasil S. Denchev, Mohammad H. Amin, Anatoly Yu Smirnov, Masoud Mohseni, and Hartmut Neven. 2016. Computational multiqubit tunnelling in programmable quantum annealers. *Nat Commun* 7 (01 2016). http://dx.doi.org/10.1038/ncomms10327

[4] Gregory J Chaitin. 1982. Register allocation & spilling via graph coloring. In *ACM Sigplan Notices*, Vol. 17. ACM, 98–105.

[5] Jianzhong Du and Joseph Y.-T. Leung. 1990. Minimizing Total Tardiness on One Machine is NP-Hard. *Mathematics of Operations Research* 15, 3 (1990), 483–495. https://doi.org/10.1287/moor.15.3.483

[6] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028 [quant-ph] (Nov. 2014). http://arxiv.org/abs/1411.4028

[7] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm Applied to a Bounded Occurrence Constraint Problem. arXiv:1412.6062 [quant-ph] (Dec. 2014). http://arxiv.org/abs/1412.6062

[8] Edward Farhi and Aram W. Harrow. 2016. Quantum Supremacy through the Quantum Approximate Optimization Algorithm. arXiv:1602.07674 [quant-ph] (Feb. 2016). http://arxiv.org/abs/1602.07674

[9] Itay Hen and Marcelo S Sarandy. 2016. Driver Hamiltonians for constrained optimization in quantum annealing. arXiv preprint arXiv:1602.07942 (2016).

[10] Itay Hen and Federico M Spedalieri. 2016. Quantum Annealing for Constrained Optimization. Physical Review Applied 5, 3 (2016), 034007.

[11] IBM. 2017. IBM Q and Quantum Computing. (2017). https://www.research.ibm.com/ibm-q/

[12] Zhang Jiang, Eleanor G. Rieffel, and Zhihui Wang. 2017. Near-optimal quantum circuit for Grover's unstructured search using a transverse field. Phys. Rev. A 95 (Jun 2017), 062317. Issue 6. https://doi.org/10.1103/PhysRevA.95.062317

[13] Eugene L. Lawler. 1977. A ''Pseudopolynomial'' Algorithm for Sequencing Jobs to Minimize Total Tardiness. Annals of Discrete Mathematics 1 (1977), 331 -- 342. https://doi.org/10.1016/S0167-5060(08)70742-8

[14] Frank Thomson Leighton. 1979. A graph coloring algorithm for large scheduling problems. Journal of research of the national bureau of standards 84, 6 (1979), 489--506.

[15] Andrew Lucas. 2014. Ising formulations of many NP problems. Frontiers in Physics 2, 5 (2014). https://doi.org/10.3389/fphy.2014.00005

[16] Masoud Mohseni, Peter Read, Hartmut Neven, Sergio Boixo, Vasil Denchev, Ryan Babbush, Austin Fowler, Vadim Smelyanskiy, and John Martinis. 2017. Commercialize Quantum Technologies in Five Years. Nature 543 (2017), 171--174. http://www.nature.com/news/commercialize-quantum-technologies-in-five-years-1.21583

[17] John Preskill. 2012. Quantum computing and the entanglement frontier. arXiv:1203.5813 (March 2012). http://arxiv.org/abs/1203.5813

[18] E. G. Rieffel and W. Polak. 2011. Quantum Computing: A Gentle Introduction. MIT Press, Cambridge, MA.

[19] Eleanor G Rieffel, Davide Venturelli, Minh Do, Itay Hen, and Jeremy Frank. 2014. Parametrized Families of Hard Planning Problems from Phase Transitions.. In AAAI. 2337--2343.

[20] E. A. Sete, W. J. Zeng, and C. T. Rigetti. 2016. A functional architecture for scalable quantum computing. In 2016 IEEE International Conference on Rebooting Computing (ICRC). 1--6. https://doi.org/10.1109/ICRC.2016.7738703

[21] Davide Venturelli, Minh Do, Eleanor Rieffel, and Frank Jeremy. 2017. Compiling Quantum Circuits to Realistic Hardware Architectures using Temporal Planners. arXiv preprint arXiv:1705.08927 (2017).

[22] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G. Rieffel. 2017. The Quantum Approximation Optimization Algorithm for MaxCut: A Fermionic View. arXiv preprint arXiv:1706.02998 (2017).

[23] Dave Wecker, Matthew B Hastings, and Matthias Troyer. 2016. Training A Quantum Optimizer. arXiv preprint arXiv:1605.05370 (2016).