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. In this work, 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.

CCS CONCEPTS

• **Mathematics of computing** → **Approximation algorithms**; • **Hardware** → **Emerging technologies**; **Quantum computation**; • **Theory of computation** → *Quantum computation theory*; *Mathematical optimization*;

KEYWORDS

quantum computing, optimization, approximation algorithms

ACM Reference format:

Stuart Hadfield[*], Zhihui Wang[+,**], Eleanor G. Rieffel[+], Bryan O'Gorman[+,†], Davide Venturelli[+,**], Rupak Biswas[+]. 2017. 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/10.1145/3149526.3149530

1 INTRODUCTION

Over the last few decades, researchers have discovered several stunning instances of quantum algorithms that provably outperform the best existing classical algorithms and, in some cases, the best possible classical algorithm [20]. 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 analytically 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, or demonstrating their effectiveness in practical applications. As prototype quantum hardware emerges, this approach to algorithm design becomes available for the evaluation of quantum heuristic algorithms.

For several years now, special-purpose quantum hardware has been used to explore one quantum heuristic algorithm, quantum annealing. Emerging gate-model processors, which are universal in that, once scaled up, they can run any quantum algorithm, will enable investigation of a much broader array of quantum heuristics beyond quantum annealing. Within the last year, IBM has made available publicly through the cloud a 5-qubit gate-model chip [13], and announced recently an upgrade to a 17-qubit chip. Likewise, Google [3] and Rigetti Computing [22], anticipate providing processors with 40–100 qubits within a year or two [18]. Many academic groups, including at TU Delft and at UC Berkeley, have similar efforts. 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 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 becomes available. The immediate question is "What experiments should we prioritize that will give us insight into quantum heuristics?" One leading candidate is QAOA.

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 [14, 23, 25, 26]. 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.

PMES'17, November 2017, Denver, Colorado, USA

ACM ISBN 123-4567-24-567/08/06…$15.00

https://doi.org/10.1145/3149526.3149530

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 is that, like quantum annealing, it is relatively straightforward 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 QAOA, 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 operators that are restricted to the feasible subspace and thus do not explore infeasible solutions. We then provide a handful of detailed examples, focusing on optimization versions of a variety of scheduling problems, which enable us to illustrate a variety of mappings, which, as we discuss briefly in the conclusions, are applicable to a large number of other combinatorial optimiztion 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 was originally introduced as an algorithm for approximate optimization, but the corresponding circuits have since been employed in other ways. QAOA circuits iteratively alternate between applications of an operator corresponding to the optimization cost function and a mixing step used to explore the solution space. In order to be able to refer to such circuits as such, we propose in [10], as part of an extension of the framework, a re-working the acronym 'QAOA' to stand for 'Quantum Alternating Operator Ansatz'; that is, to recast it as a model for the set of states representable as the application of alternating sequence of parameterized classical and mixing operators to a "simple" initial state. This has the added benefit of removing the redundancy in the commonly used expression 'QAOA algorithm'. We will refer to the circuits that implement the QAOA model as 'QAOA circuits', which we define formally in Sec. 3.2. Here, we motivate this work by giving an overview of prior results on QAOA.

Farhi *et al.* [6] proposed 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. The authors [7] were able to obtain bounds for a single iteration QAOA_1 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, or with more repetitions and well-chosen parameters, could potentially beat the approximation ratio of this classical algorithm, but proving so is challenging.

Since Farhi *et al.*'s original work, QAOA circuits have also been applied for exact optimization [14, 26] and sampling [8]. Wecker *et al.* [26] explored learning parameters for QAOA circuits on instances of MAX-2-SAT that result in high overlap with the optimal state. Jiang *et al.* [14] demonstrated that QAOA is powerful enough to obtain the $\Theta(\sqrt{2^n})$ query complexity on Grover's problem, and also provided the first algorithm within the QAOA framework to show a quantum advantage for a finite number of iterations larger than one. 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, even those with just a single iteration ($p = 1$). Their results suggest that QAOA circuits applied to sampling are promising candidates for early demonstrations of "quantum supremacy" [2, 19]. It remains an open question whether QAOA circuits provide a quantum advantage for approximate optimization.

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 $C : \{0,1\}^n \to \mathbb{R}$, where n is 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.

The goal is to find a bit string that achieves the maximum $C_{\max}$ of the cost function over the feasible subset. In practice, it often suffices to find a feasible solution that comes close to the maximum. A feasible bitstring $\mathbf{x}$ is said to be an r-approximate solution if

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

An algorithm is said to be an r-approximation algorithm if it is guaranteed to find an r-approximate solution.

As with the much-explored adiabatic quantum optimization (AQO), QAOA circuits 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, such as [20], or a review article, for an introduction to this topic.

3.1 A brief review of quantum computing

The state of a qubit, a quantum bit, is an element of a 2-dimensional complex vector space. The state vector is normalized to have unit modulus, and states that differ only by a complex multiplicative constant ("global phase") are physically indistinguishable. In quantum mechanics, Dirac's bra-ket notation is used, where $|v\rangle$ is notation for a column vector, and $\langle v|$ is notation for its Hermitian conjugate, a row vector. Two orthogonal vectors are chosen as the computational basis, $\{|0\rangle, |1\rangle\}$, to correspond to classical bit values. We use the conventional representations $|0\rangle = \begin{pmatrix}1 & 0\end{pmatrix}^\intercal$ and $|1\rangle = \begin{pmatrix}0 & 1\end{pmatrix}^\intercal$.

Multiple qubit spaces combine via the tensor product. States of multiple qubits are unit-modulus vectors in this tensor product space. 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, $|\mathbf{b}\rangle = |b_n b_{n-1} \dots b_1\rangle = |b_n\rangle \otimes \dots \otimes |b_1\rangle$, where $\mathbf{b} = (b_i)_{i=1}^n \in \{0,1\}^n$.

For convenience we will also use the label $|x\rangle$ to refer to the state vector labeled with the binary string corresponding to a non-negative (and implicitly bounded) integer x. Every n-qubit state $|\psi\rangle$ can be written as a *superposition* of such computational basis states: $|\psi\rangle = \sum_{x=0}^{N-1} a_x|x\rangle$, where $N = 2^n$ is the number of n-bit strings and the coefficients $\{a_x\}$ 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 factor* $e^{i\theta}$. Normalization requires that $\sum_{i=0}^{N_1} |a_x|^2 = 1$. The amplitudes encode measurement probabilities. In particular, measuring $|\psi\rangle$ in the computational basis gives the single bit string $\mathbf{x}$ with probability $|a_\mathbf{x}|^2$.

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.

Commuting operators have a shared eigenbasis, so to induce transitions (mixing) between the orthogonal states of the computational basis (which are eigenstates of the Z operator), X or Y operators are required. 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 correspond to "classical" Hamiltonians. For example, $Z_1X_3 = Z \otimes I \otimes X$ 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. For a Hamiltonian H that is both unitary and Hermitian, $e^{i\theta H} = \cos(\theta)I + i\sin(\theta)H$. As we will see, this property is useful for both calculations and developing intuitions. For $\theta = \pi/2$, $e^{i\theta H}$ is just H up to an irrelevant global phase of i.

3.2 A brief review of QAOA circuits

To represent the cost function $C(x)$ as a problem Hamiltonian H_C, we follow the standard practice of substituting $x_i \mapsto \frac{1}{2}(I - Z_i)$ 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 $Z_i|x_i\rangle = (-1)^{x_i}|x_i\rangle$, $\frac{1}{2}(I - Z_i)|x_i\rangle = x_i|x_i\rangle$ and thus $H_C|x\rangle = C(x)|x\rangle$. We will refer to the subspace spanned by $\{|\mathbf{x}\rangle\}$, where $\mathbf{x}$ lies in the feasible subset, as the *feasible subspace* of states.

Level p QAOA circuits, $QAOA_p$, iterate p times. On iteration i two Hamiltonians are applied sequentially,

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

The first achieves *phase separation*, with each computational basis state $|\mathbf{x}\rangle$ being multiplied by a phase factor that depends on the associated cost $C(\mathbf{x})$. The second *mixes amplitude* between computation basis states. For optimization problems, the idea is to choose β_i and γ_i such that quantum interference results in a concentration of probability amplitude on configurations x with $C(x)$ large. Repeatedly running the algorithm with those parameters prepares a quantum state which when measured in the computational basis yield an approximate solution with high probability. Higher p generally yields better solutions.

Here, we discuss the design of phase separation and mixing operators. The resulting algorithms will have to be simulated for small cases and ultimately run on emerging hardware in order to experiment with parameters and to evaluate the effectiveness of the approach. Further, there isn't a single metric when comparing the performance to other approaches, classical or quantum. Factors that can be compared include the resources required for implementation (qubits, connectivity, compilation overhead [23], and the running time, both per-level and the number of levels), the probability of achieving a good approximation ratio, and the ease with which good parameters can be found, either analytically [25] or by performing a search with the quantum computer [9]. When QAOA is used for purposes other than approximation, different metrics may be used. For example, for sampling applications, one may need to evaluate properties of the output distribution.

3.3 Framework for mapping combinatorial optimization problems to QAOA

All problem and instance 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 setting), all problem dependence must be captured in the cost Hamiltonian on such devices. 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 produces mappings for an optimization problem that work with any mixing Hamiltonian, and thus a uniform mixing Hamiltonian can be implemented universally for all problems, as is done on current quantum annealers.

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. 4.1. Here, we provide a general framework, taking inspiration from the example developed in Sec. VII of Farhi *et al.* [6], and building on a theory developed for AQO by Hen & Spedalieri [12] and Hen & Sarandy [11].[1]

In the QAOA circuit setting, some of the design criteria for AQO in [12] are not needed, enabling us to have greater flexibility in the design of the mixing Hamiltonian.[2] With QAOA as with AQO,

[1] Conventionally, AQO seeks to minimize a cost function, whereas QAOA seeks to maximize; the corresponding Hamiltonians differ only by a sign.

[2] 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 be concerned with the gap between the ground and first excited subspaces.

constraining the search to feasible states may be more effective than imposing penalty terms for two reasons. First, the feasible subspace may be smaller than the full space and so general mixing terms are likely to have difficulty with ensuring a high probability of getting a feasible solution at the end of the algorithm. Second, the large weight that must be placed on penalty terms for them to serve their purpose effectively rescales the cost function, reducing the possible phase separation between feasible states. Quantifying these concerns and analyzing the tradeoff between the improved effectiveness of the algorithm and the greater difficulty of implementation is an important direction for future work.

The mixing Hamiltonian should both restrict exploration to the feasible subspace and promote exploration of that subspace. Motivated by [11], the mixing Hamiltonian must

- preserve the feasible subspace, so that for all angles the resulting unitary takes feasible states to feasible states, and
- provide state transitions sufficient for the resulting unitary to mix any two feasible states (for some angles).

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, though this is not a necessary condition. 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 feasible subspace.

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.* [12] explore, in the AQO setting, terms of the form $\mathrm{SWAP}_{ij} = \frac{1}{2}(I + X_iX_j + Y_iY_j + Z_iZ_j)$. When applied to two-qubit computational basis states on qubits i and j, it takes $|0,1\rangle_{i,j}$ to $|1,0\rangle_{i,j}$ and vice versa, while leaving $|0,0\rangle_{i,j}$ and $|1,1\rangle_{i,j}$ unchanged. Thus, the operation swaps the bit values of the two qubits in the computational basis, generalizing the classical SWAP operation. Note that $\mathrm{SWAP}_{i,j}$ is both unitary and Hermitian. The more general unitary $e^{i\theta\mathrm{SWAP}_{ij}} = \cos(\theta)I + i\sin(\theta)\mathrm{SWAP}_{ij}$ is a combination of the identity and $\mathrm{SWAP}_{i,j}$; in particular it applies $\mathrm{SWAP}_{i,j}$ exactly for $\theta = \pi/2$. The SWAP operation preserves the Hamming weight of bit strings. We will use it to incorporate various hard constraints, such as requiring a fixed Hamming weight, into mixing Hamiltonians. Because 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 OPTIMIZATION PROBLEMS TO QAOA

Here we apply the framework of Sec. 3 to design mappings for five optimization problems. We chose these problems to show diversity in the mappings and in the techniques used for mappings. The first three are different optimization versions of vertex coloring. We also give mappings for the traveling salesman problem and single machine scheduling. As we comment in the conclusions, a number of other combinatorial optimization problems have similar mappings.

4.1 Graph coloring optimization problems

Graph-k-Coloring is an important NP-complete (for $k \geq 3$) problem with many applications, such as scheduling [16, 21] and memory allocation [4] problems. In this work, we are concerned with *vertex colorings*. 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 (is adjacently to differently colored vertices). Graph coloring corresponds to a scheduling problem in which the colors correspond to time slots, the vertices correspond to tasks, and edges between tasks correspond to resource conflicts, i.e. both task require the same resource so cannot be scheduled for the same time slot.

Several optimization variants of Graph-k-Coloring are known; we focus on the following three:

- **MaxColorableSubgraph.** Given a graph and a number of colors k, find a color assignment such that the number of properly colored edges is maximized.
- **MaxColorableInducedSubgraph.** Given a graph and a number of colors k, determine the maximum number of vertices for which all edges in the graph can be properly colored.
- **MinGraphColoring.** 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 and the second of Independent Set. As we will see, there exists a mixing Hamiltonian for the first that has a relatively simple form that depends only on the problem size, i.e. is instance-independent. The mixing terms we introduce for the latter two problems are more involved and instance-dependent.

4.1.1 MaxColorableSubgraph. This optimization version of graph coloring is analogous to maximum satisfiability, where instead of requiring that all constraints are satisfied we seek to maximize the number of satisfied constraints.

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 one-hot encoding (Hamming weight 1 bitstrings in which the single 1, the "hot" bit, indicates the color assigned to the the vertex). This encoding, which uses k qubits per vertex, has been used in quantum annealing [11, 12, 17, 21]. 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. As an initial state, we can take all vertices colored the same color.

For each vertex, a hard constraint is that the vertex be assigned exactly one color, i.e.,

$$\sum_{i=1}^{k} x_{v,i} = 1. \tag{2}$$

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

$$m - \sum_{\{u,v\}\in E} \sum_{i=1}^{k} x_{u,i} x_{v,i}, \tag{3}$$

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

$$H_C = \frac{km}{4} - \frac{1}{4} \sum_{\{u,v\}\in E} \sum_{i=1}^{k} Z_{u,i} Z_{v,i} \ . \tag{4}$$

We can rewrite the constraint Eq. (2) as

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

so that the translated Hamiltonian

$$H_A = -\frac{1}{4} \sum_{v} \left(2(k-2) \sum_{i=1}^{k} Z_{v,i} - \sum_{i\neq j}^{k} Z_{v,i} Z_{v,j} \right) \tag{5}$$

admits the feasible subspace as its ground subspace.

We seek a mixing Hamiltonian H_M, that meets the criteria laid out in Sec. 3, keeping the evolution within the feasible subspace. For each vertex v, we include a term of the form

$$B_v = \frac{1}{k} \sum_{i=1}^{k} X_{v,i} X_{v,i+1} + Y_{v,i} Y_{v,i+1}, \tag{6}$$

with indices taken modulo k, 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, each term generates transitions between colors on vertex v. This mixer can be efficiently compiled to a quantum circuit [24]. A version of B_v with more connectivity provides stronger mixing, though at the cost of requiring more resources for compilation. We will explore this tradeoff in future work.

Directly extending the MaxCut application of [6], the color of each vertex could be encoded in binary using $\lceil \log_2 k \rceil$ qubits. While such an encoding efficiently represents the required states, the problem Hamiltonian would be much more complicated, requiring more complex operators 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.

4.1.2 MaxColorableInducedSubgraph. 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.4.1.1, with variables $x_{v,1}, \ldots, x_{v,k}$, but with one additional variable $x_{v,0}$ 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 subgraph induced by 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:

$$C = m - \sum_{v} x_{v,0}, \tag{7}$$

which seeks to maximize the number of colored vertices by minimizing the number of uncolored ones. The variables $x_{v,0}$ are not included in the sum since they correspond to uncolored nodes that are not part of the induced subgraph. The corresponding problem Hamiltonian, ignoring constants, is

$$H_C = \frac{1}{2} \sum_{v} Z_{v,0} \tag{8}$$

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

$$(\bar{x}_{v_1,i} \bar{x}_{v_2,i} \ldots \bar{x}_{v_{D_u},i}) \cdot \mathrm{SWAP}(x_{u,0}, x_{u,i}), \tag{9}$$

where $v_1, \ldots, v_{D_u}$ are the 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

$$B_{u,i} = \frac{1}{2^{D_u}} (X_{u,0} X_{u,i} + Y_{u,0} Y_{u,i}) \prod_{j=1}^{D_u} (I + Z_{v_j,i}). \tag{10}$$

The overall mixing Hamiltonian is $H_M = \sum_u \sum_i B_{u,i}$. Since H_M 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. A possible initial state is the one in which all vertices are uncolored.

4.1.3 MinGraphColoring. **Problem:** Given a graph $G = (V, E)$, minimize the number of colors required to properly color it.

We take as the feasible subspace proper $k = D_G + 2$ colorings (some colors may not be used). Any proper $D_G + 2$ coloring (easy to find classically) can serve as an initial state. With $k = D_G + 2$ colors, single vertex recoloring allows transitions between any two feasible states within the feasible space; any coloring using at most $D_G + 1$ colors can be transformed into any other coloring using at most $D_G + 1$ colors by a series of moves that changes the color of one vertex at a time while maintaining a proper coloring at each step.

For the mixing term, we use a controlled operation similar to Eq. (9) in Sec. 4.1.2 but that allows the re-coloring of a vertex only when doing so results in a proper coloring. 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 function

$$C = \sum_{j=1}^{D+2} \prod_{u \in V} \bar{x}_{u,j}. \tag{11}$$

The corresponding problem Hamiltonian is

$$H_C = \frac{1}{2^n} \sum_{j=1}^{D+2} \prod_{u \in V} (I + Z_{u,j}). \tag{12}$$

4.2 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 $d_{u,v} \in \mathbb{R}$, 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 $\sum_{j=1}^{n} x_{v,j} = 1$ (each v visited exactly once), and for each position j have $\sum_{v \in V} x_{v,j} = 1$ (a single vertex visited at each stop). An arbitrary ordering of all vertices can be used as the initial state. The objective function is

$$\sum_{\{u,v\} \in E} d_{u,v} \sum_{j=1}^{n} \left(x_{u,j} x_{v,j+1} + x_{v,j} x_{u,j+1} \right) \tag{13}$$

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 $H_M = \sum_{u<v} B_{u,v}$,

$$B_{u,v} = \prod_{j=1}^{n} \mathrm{SWAP}((u,j),(v,j)) \tag{14}$$

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

$$H'_M = \sum_{i \neq j} \sum_{u \neq v} |0_{ui} 1_{u,j} 1_{vi} 0_{v,j}\rangle\langle 1_{ui} 0_{u,j} 0_{vi} 1_{v,j}| \tag{15}$$

$$= \sum_{i \neq j} \sum_{u \neq v} S^-_{u,i} S^-_{v,j} S^+_{u,j} S^+_{v,i} \ . \tag{16}$$

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

4.3 Single Machine Scheduling (SMS)

We consider the SMS problem to minimize the total tardiness [5, 15].

Problem: Given n jobs, each with a running time p_j, a deadline d_j, and a weight w_j, to run on a single machine that takes one job at a time, find a schedule that minimizes the total weighted tardiness $T = \sum_{j=1}^{n} T_j$, where $T_j \geq 0$ gives the amount of time job j is late. In the three-fields notation in scheduling, the problem is specified as $(1|d_j|\sum w_j T_j)$. All parameters are taken to be integers.

For each job j, we use a binary variable $x_{j,t}$ to indice whether or not job j starts at time t, where $t = 0, \ldots, h$, and $h = \sum_j p_j - \min_j\{p_j\}$. Feasible strings are those for which each job is assigned a single start time, i.e. $\sum_t x_{j,t} = 1$, and there are no overlapping jobs. Any ordering of the jobs, with each job starting after the sum of the previous jobs' processing times is a feasible initial state. The cost function, i.e., the weighted total tardiness becomes

$$C = \sum_j w_j \sum_{(d_j - p_j) < t < h} x_{j,t}(t + p_j - d_j), \tag{17}$$

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.

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

$$H_M = \sum_{j \neq j'} \sum_{t=0}^{h} S^-_{j,t} S^-_{j',t+p_j} S^+_{j,t+p_{j'}} S^+_{j',t} + h.c. \ . \tag{18}$$

5 CONCLUSIONS

This work is the starting point of a larger research project to construct QAOA mappings for other problems and analyze their effectiveness. In [10], we provide mappings for a variety of other problems, extending the techniques developed here. The problems graph partitioning, maximum bisection, maximum vertex-k-cover, edge coloring, minimum clique cover, and weighted versions of these problems have mappings closely related to those of Sec. 4.1, and the techniques in Secs. 4.2 and 4.3 can be extended to more generally scheduling-type problems. A controlled bit-flip mixer can be used for maximum independent set, maximum clique, minimum vertex cover, maximum set packing, and min set cover. Other problems may require new techniques.

Future directions include:

- Investigate the performance of QAOA under mixing terms with more or less connectivity, to understand tradeoffs in terms of algorithmic efficiency and ease of implementability.
- Optimizing compilations of QAOA circuits for near-term hardware architectures.
- Investigating how different initial states contribute to the efficiency of the algorithm. While any state in the feasible subspace will do, the algorithm may be more effective starting from some initial states as opposed to others. Tradeoffs in the efficiency with which such states can be implemented versus performance gains in the algorithm will be evaluated in future research, as well as the effectiveness of other prescriptions, such as random restarts initialized by random classical states.
- Further develop parameter setting strategies, and investigate how different formulations of the same problem affect the ease with which good parameters can be found.

6 ACKNOWLEDGEMENTS

The authors would like to thank the other members of NASA's QuAIL group for useful discussions about QAOA-related topics,

particularly Salvatore Mandrà and Zhang Jiang. S.H. was supported by NASA under award NNX12AK33A. He would like to thank the Universities Space Research Association (USRA) and NASA Ames Research Center for the opportunity to participate in the program that enabled this research. Z.W. and D.V. were supported by NASA Academic Mission Services, contract number NNA16BD14C. B.O. was supported by a NASA Space Technology Research Fellowship. The authors would like to acknowledge support from the NASA Advanced Exploration Systems program and NASA Ames Research Center. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright annotation thereon.

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). http://arxiv.org/abs/1505.03424

[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 (Jan. 2016).

[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.

[6] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028 (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 (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 (Feb. 2016). http://arxiv.org/abs/1602.07674

[9] Gian Giacomo Guerreschi and Mikhail Smelyanskiy. 2017. Practical optimization for hybrid quantum-classical algorithms. arXiv:1701.01450 (2017). http://arxiv.org/abs/1701.01450

[10] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor G. Rieffel, Davide Venturelli, and Rupak Biswas. 2017. From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. arXiv:1709.03489 (2017).

[11] Itay Hen and Marcelo S Sarandy. 2016. Driver Hamiltonians for constrained optimization in quantum annealing. Physical Review A 93, 6 (2016), 062312.

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

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

[14] Zhang Jiang, Eleanor G Rieffel, and Zhihui Wang. 2017. Near-optimal quantum circuit for Grover's unstructured search using a transverse field. Physical Review A 95, 6 (2017), 062317.

[15] Eugene L. Lawler. 1977. A "Pseudopolynomial" Algorithm for Sequencing Jobs to Minimize Total Tardiness. Annals of Discrete Mathematics 1 (1977), 331 – 342. http://www.sciencedirect.com/science/article/pii/S0167506008707428

[16] 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.

[17] Andrew Lucas. 2014. Ising formulations of many NP problems. Frontiers in Physics 2, 5 (2014), 1–15.

[18] 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

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

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

[21] 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.

[22] 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.

[23] Davide Venturelli, Minh Do, Eleanor Rieffel, and Frank Jeremy. 2017. Compiling Quantum Circuits to Realistic Hardware Architectures using Temporal Planners. arXiv:1705.08927 (2017). http://arxiv.org/abs/1705.08927

[24] Frank Verstraete, J Ignacio Cirac, and José I Latorre. 2009. Quantum circuits for strongly correlated quantum systems. Physical Review A 79, 3 (2009), 032316.

[25] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G. Rieffel. 2017. The Quantum Approximation Optimization Algorithm for MaxCut: A Fermionic View. arXiv:1706.02998 (2017). http://arxiv.org/abs/1706.02998

[26] Dave Wecker, Matthew B Hastings, and Matthias Troyer. 2016. Training a quantum optimizer. Physical Review A 94, 2 (2016), 022309.