QED driven QAOA for network-flow optimization

Yuxuan Zhang[1], Ruizhe Zhang[2], and Andrew C. Potter[1]

[1]Center for Complex Quantum Systems, University of Texas at Austin, Austin, TX 78712, USA

[2]Department of Computer Science, University of Texas at Austin, Austin, TX 78712, USA

We present a general framework for modifying quantum approximate optimization algorithms (QAOA) to solve constrained network flow problems. By exploiting an analogy between flow-constraints and Gauss' law for electromagnetism, we design lattice quantum electrodynamics (QED)- inspired mixing Hamiltonians that preserve flow constraints throughout the QAOA process. This results in an exponential reduction in the size of the configuration space that needs to be explored, which we show through numerical simulations, yields higher quality approximate solutions compared to the original QAOA routine. We outline a specific implementation for edge-disjoint path (EDP) problems related to traffic congestion minimization, numerically analyze the effect of initial state choice, and explore trade-offs between circuit complexity and qubit resources via a particle-vortex duality mapping. Comparing the effect of initial states reveals that starting with an ergodic (unbiased) superposition of solutions yields better performance than beginning with the mixer ground-state, suggesting a departure from the "short-cut to adiabaticity" mechanism often used to motivate QAOA.

1 Introduction

Combinatorial optimization (CO) tasks present many classically-hard computational problems, and abound in practical applications from vehicle routing to resource allocation, job scheduling, portfolio optimization, and integrated circuit layout. Finding optimal solutions to many practically relevant classes of CO problems is an NP-complete task, which is effectively intractable for large problems. In the past decades, quantum computers promise tantalizing speedups on certain classically-hard computational problems, such as integer factoring [22] and structured searching [16]. Unfortunately, barring an upheaval of complexity theoretic dogma, quantum optimization algorithms are not expected to efficiently yield optimal solutions to NP-hard problems. However, for classical optimization on hard problems, one typically aims for reasonable but suboptimal approximations, and tremendous effort has been put into improving the quality of approximate solutions. In a similar vein, there is widespread hope that quantum-heuristics could yield better approximate solutions than their classical counterparts.

This hope has been largely fueled by the introduction of the Quantum Approximate Optimization Algorithm (QAOA), a hybrid classical/quantum framework originally motivated as a variational spin-off of the Quantum Adiabatic Algorithm (QAA) [11]. QAOA consists of p-rounds of stroboscopic alternation between a classical cost Hamiltonian and a quantum mixing Hamiltonian, with time intervals for each evolution treated as variational parameters that are classically optimized. While it was initially suggested that even a single round ($p = 1$) QAOA could provide a quantum-improvement over classical state-of-the-art [10], the quantum/classical gap was quickly closed [2], and there is growing evidence [5, 12, 13, 19] that p must generically scale with the problem-size in order to achieve improved approximate solutions. Due to the difficulty of analyzing QAOA-performance at large-p, establishing rigorous evidence of quantum advantage remains elusive, and the practical value of QAOA will likely be decided empirically (like many successful classical heuristic methods).

Making QAOA into a successful quantum heuristic will require advances in problem encoding, and algorithm efficiency. A key weakness of traditional QAOA is that many relevant CO problems impose constraints among variables, which are not respected by the QAOA heuristic. A typical approach to QAOA would be to map a CO problem into a binary integer linear program (BILP), whose objective function is mapped to an Ising-like spin model that can be implemented on quantum hardware. In this formulation, constraints are typically softly enforced by adding a term to the cost Hamiltonian that energetically penalizes constraint violations. This approach is frequently inefficient, as it can result in exploration of an exponentially-large (in problem size) set of infeasible (constraint-violating) configurations, which has been shown to dramatically hamper performance [23].

An alternative technique is to modify the QAOA procedure to automatically satisfy constraints throughout the algorithm. In [17, 23], this approach was used to tackle graph-coloring problems (among others), where a number-conserving mixing Hamiltonian was designed to preserve a one-hot encoding structure. Due to the intimate connection between symmetries and conservation laws, this highlights a connection between physical symmetries and constraints in CO problems, and suggests that physics-inspired solutions may be fruitful.

In this work, we exploit another common "symmetry" found in physical systems: gauge-invariance [1], to implement a constraint-satisfying mixer for network flow problems. Network flow problems are defined on graphs, where each link of a graph has a directed flow of "goods" that takes real or integer values. In practice, flow could represent an amount of vehicles, goods, communication packets, etc., being transported through the network. Real-valued flow problems tend to admit classically efficient solutions via linear programming, whereas multi-commodity integer flow problems are often classically hard. Integer flow problems have a wide array of applications from vehicle routing, traffic congestion minimization, and package delivery, to communication network optimization. Each of these problem formulations share a common constraint structure: the amount of flow entering a vertex must match the total outgoing flow, plus (minus) a fixed amount at certain source (sink) nodes.

This flow structure is a discrete analog of Gauss law in electromagnetism: $\nabla \cdot \boldsymbol{E} = \rho$, where ρ is the charge density, if we re-interpret the electric field $\boldsymbol{E}$ as a flow emanating out of a node, and the charge ρ as the amount of sourced or sinked goods. The central idea of this paper will be to exploit this analogy to develop a lattice quantum electrodynamics (QED) inspired QAOA-mixer that automatically preserves network-problem flow constraints.

The structure of the paper is organized as follows: we briefly summarize QAOA from the generalized perspective advocated in [10], and review the structure of lattice-QED. We then establish a direct relationship to flow problems on finite-dimensional graphs, and define a constraint-preserving generalization of QAOA using a QED-style mixer. We numerically compare the performance of modified QED-QAOA and the original (X-mixer) QAOA on a (classically easy) flow maximization problem, and show that the quality of approximate solutions increases in a way that is consistent with exponential-in-problem size scaling. We then explore QED-mixer performance on classically-hard traffic congestion minimization problems, and study the behavior with increasing problem size and number of QAOA rounds. A key step in the algorithm is preparing an initial constraint-preserving state that is a quantum superposition including all constraint-preserving states. Unlike the original QAOA, where the X-mixer ground-state can be accomplished with a transversal set of single-qubit rotations, the QED-mixer ground-state is more complicated. We explore and compare multiple strategies for initial state preparation, and find, perhaps surprisingly, that the QED-mixer ground-state is not optimal, suggesting a departure from the adiabatic-algorithm reasoning often used to motivate QAOA.

[1]Strictly speaking, gauge invariance is not an ordinary symmetry, however the analogy is frequently useful.

2 Quantizing network flow problems

To set the stage, we briefly review the constraint structure of network flow problems, introduce the specific problem types that we will use to illustrate the QED-inspired QAOA approach, and describe an implementation of their cost function as a quantum Hamiltonian acting on qudits.

2.1 Constraints in Flow Problems

A flow problem is defined on a graph $\mathcal{G}$ with vertices $\mathcal{V}$ and edges $\mathcal{E} = \{(u,v)|\ u,v \in \mathcal{V}$ are connected$\}$. Here $\mathcal{G}$ is required to be a directed graph by many versions of flow problems, such as max-flow problems, but can be undirected in other cases like EDP. We denote the total number of vertices as $|\mathcal{V}|$, and the number of edges as $|\mathcal{E}|$. On each edge, we define a flow: $f(u,v) \in \mathbb{F}$ taking value in some field $\mathbb{F}$, with $f(u,v) = -f(v,u)$. To facilitate implementation on discrete-leveled quantum computing systems, in this paper we will specialize to integer flows of k-different commodities (i.e. $\mathbb{F} = \mathbb{Z}^k$). We define the vertex from which a commodity originates or terminates as a source or sink node respectively. We denote the sets of source and sink nodes as $\{s_i\}_{i=1}^k$ and $\{t_i\}_{i=1}^k$, and the amount of flow to be delivered for the i^{th} source-sink pair as d_i.

While there are a large variety of flow-problem formulations, they all share a common constraint structure. Namely, valid flows may begin and terminate only on source and sink nodes, respectively:

$$\sum_{v:(u,v)\in\mathcal{E}} f_i(u,v) = d_i(\delta_{u,s_i} - \delta_{u,t_i}) \quad \forall\ u \in \mathcal{V}. \qquad (1)$$

Figure. 1 illustrates selected examples of valid and invalid flow configurations.

In addition, many flow problems impose additional capacity constraints on how many of each type of commodities may flow through a particular edge:

$$\sum_{i\in[k]} |f_i(u,v)| \leq c(u,v) \quad \forall\ (u,v) \in \mathcal{E}, \qquad (2)$$

where $c(u,v) \in \mathbb{Z}_+$ is referred to as the edge-capacity: the total amount of all types of flows cannot exceed the capacity on that edge.

Flow problems come in many varieties. Some, such as the single-commodity max flow problem, have efficient classical algorithms. However, many practical problems require introducing multiple commodi-

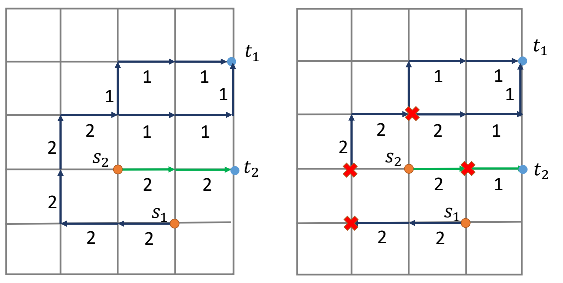


Figure 1: **Example flows on a** 5×5 **grid graph** A feasible network flow configuration (L) and an unfeasible configuration(R): the arrows stand for flow directions, and different flows are distinguished by colors with numbers representing the amount of flow on each edge(a certain assignment of the flows in the graph is called a *configuration*).

ties and imposing finite edge-capacities, which typically results in hard optimization problems. For example, the problem of maximizing capacitated integer flow was proven to be NP-complete even for only two source-sink pairs [9].

2.1.1 Qudit encoding

To encode integer flow problems onto quantum hardware, we imagine using a register of $(2d_i+1)$-level qudits (possibly encoded into ordinary qubits using, e.g. binary or one-hot encoding) for each commodity and each edge $(u,v)\in\mathcal{E}$, with the qudit computational basis states $\{|-d_i\rangle,\dots,|-1\rangle,|0\rangle,|1\rangle\dots|d_i\rangle\}$ indicating the amount of flow on that link [2] We note that, for this encoding the dimension of the entire Hilbert space is thus the same as the number of all possible configurations on the graph, which is $\prod_i(2d_i+1)^{|\mathcal{E}|}$.

The total Hilbert space of this encoded system contains exponentially many infeasible configurations that violate the flow constraints (Eq. (1)). The precise ratio of feasible (flow-conserving) to infeasible (flow-violating) solutions varies by graph; however, it is generally exponentially small in $|\mathcal{V}|$. To see this, note that, the distance between a pair of randomly chosen source and sink points is typically poly($|\mathcal{V}|$), and for each valid path from source to sink, removing any edge along the path from source to sink would result in an infeasible solution, resulting in combinatorially many infeasible solutions for each feasible one.

[2]One could partially enforce capacity constraints by restricting the basis to $\{|-c(u,v)\rangle\dots|c(u,v)\rangle\}$ for each commodity, however, this choice would conflict with our scheme for producing a valid initial state.

2.1.2 Flow operators

We also introduce quantum flow operators on each edge $e\in\mathcal{E}$, and for each commodity type $i=1\dots k$:

$$E_e^{(i)}=\sum_{f=-d_i}^{d_i} f|f\rangle\langle f|_e\otimes\mathbb{1}_{e'\neq e} \tag{3}$$

where the symbol E anticipates an analogy with electric field operators in lattice-QED. Applying $E_e^{(i)}$ on a state would just return the amount of flow on edge e.

Furthermore, we denote the operator whose eigenstates are equal weighted superpositions of flow values as:

$$X_e^{(i)}=\sum_{f,f'=-d_i}^{d_i}|f'\rangle\langle f|_e\otimes\mathbb{1}_{e'\neq e} \tag{4}$$

which is the natural qudit analog of the Pauli-X operator.

We also define the total flow of all goods on edge $e\in\mathcal{E}$, as $E_e\equiv\sum_{i=1}^k E_e^{(i)}$, and similarly $X_e\equiv\sum_{i=1}^k X_e^{(i)}$. The conventional QAOA mixer is built from $H_M=-\sum_e X_e$, which indiscriminately mixes between feasible and infeasible solutions, and has a tendency to get "lost" in the exponentially larger infeasible parts of Hilbert space.

2.2 The Edge-Disjoint Path Problem

The main problem we will consider in this paper is a variant of traffic congestion minimization problem known as the edge-disjoint paths problem (EDP), often regarded as a particularly clean problem that characterizes the NP-hardness of flow optimization. Qualitatively, the frequently studied optimization version of EDP seeks to route k different commodities without "congestion", i.e., without multiple commodities flowing through the same edge:

EDP: *Given a undirected graph $G(V,E)$ with k source/sink-pairs (s_i,t_i), find k paths connecting s_i and t_i for all $i\in[k]$ such that the maximum of congestion in each edge is minimized.*

Since the maximum of congestion is a global function that is hard to implement on a quantum circuit, we can reformulate EDP's cost function by locally penalizing all congested edges instead:

$$\min\quad C\equiv\sum_{(u,v)\in E}\max\left\{0,\ \sum_{i\in[k]}|f_i(u,v)|-1\right\}\quad s.t.$$
$$\sum_{v:(u,v)\in\mathcal{E}} f_i(u,v)=d_i(\delta_{u,s_i}-\delta_{u,t_i})\quad\forall\,u\in\mathcal{V}. \tag{5}$$

In other words, in this version one aims at minimizing the total amount of congestion on all edges instead of the maximum. Notice that an optimal solution with $C = 0$ will still be a solution of the EDP problem (with no congestion), whereas $C > 0$ configurations may be regarded as approximate solutions of the relaxed EDP.

EDP has been shown to be NP-hard even with a rather modest scaling of commodity types ($k \sim \log |\mathcal{V}|$) [6]. We restrict our attention to EDPs on planar graph, where the problem remains NP-hard [6].

To convert the EDP cost-function into a quantum Hamiltonian, we reformulate the cost function into an analytic form, and introduce the EDP cost Hamiltonian (using the encoding described above):

$$H_{\mathrm{C,EDP}} = \sum_{e\in E} [\frac{(2E_e^2 - 1)^2 - 1}{48}] \tag{6}$$

which has vanishing energy for non-congested links (with $E_e = 0, 1$) and penalizes higher congestion. The normalization is chosen such that minimally congested links with $E_e = \pm 2$ have one unit of energy cost. Eq. (6) is a reformulation of the cost function in Eq. (5), whose flow constraints will be dealt with in later sections.

2.3 The Single Source Shortest Path Problem

For classical simulations, the fully unconstrained multi-commodity Hilbert space quickly becomes intractable. Therefore, to benchmark the modified QAOA performance against the original formulation, we also consider a much simpler class of single source shortest path problem (SSSP), which seek the shortest path (on a weighted graph) between a single source and sink with unit demand ($d = 1$):

SSSP: *Given a weighted undirected graph $G(\mathcal{V}, \mathcal{E})$, with weights $\{w_e : e \in \mathcal{E}\}$, and a single pair of source and sink vertices $s, t \in \mathcal{V}$, find the minimal length path connecting s and t where length is defined as the sum of the weights along the path.*

Notice that SSSP can be defined on either directed, undirected or mixed graphs, and we choose to study the undirected version for consistency with the study of EDP. Efficient classical algorithms for SSSP [3, 8] are textbook-standard materials (see also [20] for a quantum algorithm for directed acyclic graphs). In this work, we do not aim to improve solution of SSSP, but only to use this problem as a benchmark to compare the performance of different QAOA mixers in the graph routing problem. Importantly, none of the QAOA strategies we test take advantage of the classically efficient solution, providing a fair comparison.

Since SSSP is a direct analogy of EDP (at $k = 1$), we can use the same encoding scheme and write the

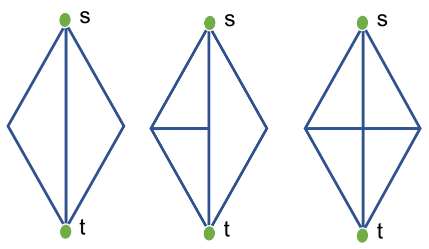


Figure 2: **Triangle graphs used in the study of SSSP problem** In SSSP simulations we look for the shortest (lowest-weight) path from the top node to the bottom node, where weight on each edge is randomly assigned.

# of Triangles	Total # States	# Feasible States	Feasible fraction
2	729	3	4.1×10^{-3}
3	2187	4	1.8×10^{-3}
4	6561	8	1.2×10^{-3}

Table 1: **A comparison between total and feasible Hilbert space dimension** Total Hilbert space dimension ($|\mathcal{H}_{\mathrm{tot}}|$) and feasible sub-space dimension ($|\mathcal{H}_{\mathrm{f}}|$), and ratio of feasible to total states $|\mathcal{H}_{\mathrm{f}}|/|\mathcal{H}_{\mathrm{tot}}|$.

cost Hamiltonian as

$$H_{\mathrm{C,SSSP}} = \sum_{e\in\mathcal{E}} w_e (E_e)^2 \tag{7}$$

where w_e denote the edge e's weight. Since only a single type of flow with demand 1 presents in the problem, the valid flow values are just -1,0,1 on any edge. The size of the Hilbert space of SSSP is thus $3^{|\mathcal{E}|}$, which, for large graphs, is far less than the $3^{k|\mathcal{E}|}$ (with $k \geq 2$) required for EDP, allowing us to classically simulate relatively larger instances.

With these problem classes in hand, we now turn to the task of modifying the QAOA algorithm to preserve the network flow constraints, beginning with a brief review of QAOA to set notation.

3 From QAOA to Lattice QED

QAOA is designed to sample from low-cost states of a cost Hamiltonian H_C which is diagonal in the computational basis, and represents the objective function of the optimization problem in question. In its original incarnation [10], the initial state $|\psi_0\rangle$, is chosen to be the ground-state of a mixing Hamiltonian $H_M = H_{M,X}$ with:

$$H_{M,X} = -\sum_i X_i, \tag{8}$$

which we will refer to as the "X-mixer." Subsequent generalizations [17] considered more complicated forms of H_M designed to preserve constraints of various forms. The algorithm proceeds by evolving:

$$|\psi_p(\boldsymbol{\gamma},\boldsymbol{\beta})\rangle = \prod_{j=1}^{p} e^{-i\beta_j H_M} e^{-i\gamma_j H_C}|\psi_0\rangle \tag{9}$$

to generate a variational wave function characterized by real-parameters $\{\gamma_i\}$ and $\{\beta_i\}$ $(i = 1, 2, ...p)$, which are classically optimized (using the classical routine of ones choice) to minimize the expected cost: $\boldsymbol{\gamma}_*, \boldsymbol{\beta}_* = \arg\min \varepsilon_C$, where:

$$\varepsilon_C \equiv \langle\psi_p(\boldsymbol{\gamma},\boldsymbol{\beta})|H_C|\psi_p(\boldsymbol{\gamma},\boldsymbol{\beta})\rangle. \tag{10}$$

This biases the wave-function amplitude of $|\psi_p(\boldsymbol{\gamma}_*,\boldsymbol{\beta}_*)\rangle$ towards low-cost configurations, such that repeated sampling from this state preferentially yields low-cost solutions.

In the limit of infinite p, QAOA contains Quantum Adiabatic Algorithm (QAA) as a subset of possible solutions and is guaranteed to find the exact optimum. For hard problem instances, precisely following the adiabatic path may require p to grow superexponentially with problem-size, but it is hoped that approximate short-cuts to this adiabatic solution may be variationally identified with far lower p.

To apply this formalism, one must first map the optimization problem variables onto qubits, such that the cost for each qubit configuration can be computed in a local manner. For constrained optimization problems, this often results in a wasteful encoding in which many qubit states do not satisfy the feasibility constraints. One possible strategy would be to energetically penalize the constraint violation by introducing a penalty term into H_C for unsatisfied constraints. While straightforward in its implementation, this strategy results in wasteful exploration of (typically exponentially many) configurations corresponding to infeasible solutions, degrading algorithm performance. An alternative option [17] is to identify an alternate mixing term H_M which automatically preserves the constraint structure. Then, if an initial state can be prepared to satisfy all constraints, the algorithm will only search inside the feasible subspace. In what follows, we focus on the constraints common to a large variety of network flow problems and show how to encode them into an appropriate constraint-preserving mixer inspired by lattice-QED.

3.1 Lattice QED Hamiltonian

The flow constraints described in Eq. (5) are of precisely the same form as Gauss's law for lattice-QED, if we interpret each commodity flow as a different "flavor" of electric field, and the corresponding sources and sinks as positive and negative charge d_i. This suggests that we can use gauge-invariant lattice-QED Hamiltonians to implement constraint-preserving mixers for the network flow QAOA. Here, we briefly review some relevant lattice-QED notations and formalisms. In what follows, we specialize them to planar graphs, although our construction generalizes to arbitrary finite-dimensional graphs (but would become infeasible for fully-connected graphs). For notational simplicity, we initially suppress the commodity ("flavor") label.

The Hamiltonian formulation of the (compact) lattice QED, on a planar graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is defined by introducing discrete analogs of the continuum electric field $\boldsymbol{E}(\boldsymbol{r})$ and vector potential $\boldsymbol{A}(\boldsymbol{r})$. Specifically, a (gauge-redundant) Hilbert space is defined by electric field operators $E_{uv} = -E_{vu}$ on each edge $(u, v) \in \mathcal{E}$ whose eigenstates are denoted $|e_{uv}\rangle$ with $e_{uv} \in \mathbb{Z}$. Electric fields are oriented such that $E_{vu} = -E_{vu}$. The conjugate operator to E_{uv} is denoted by $e^{-iA_{uv}}$, which raises or lowers the electric field:

$$e^{iA_{uv}} E_{uv} e^{-iA_{uv}} = E_{uv} + 1, \tag{11}$$

and $[e^{-iA_{uv}}, E_{wx}] = 0$ for $(w, x) \neq (u, v)$.

Physical states are defined by projecting onto subspace that satisfies a lattice analog of the continuum Gauss' law $\nabla\cdot\boldsymbol{E}(\boldsymbol{r}) = \rho(\boldsymbol{r})$, i.e. $\sum_{u:(u,v)\in\mathcal{E}} E_{uv} = \rho_u$, which is precisely the same form as the flow constraint (Eq. (1)), provided that we equate the electrical charge with demand, d. The Gauss' law is equivalent to demanding invariance under gauge transformations:

$$e^{-iA_{uv}} \to e^{-i\phi_u} e^{-iA_{uv}} e^{i\phi_v} \tag{12}$$

$$|\psi\rangle \to e^{i\sum_{u\in\mathcal{V}}\phi_u\rho_u}|\psi\rangle \tag{13}$$

for any vertex-dependent phases $e^{i\phi_v} \in U(1)$.

A special role is played by gauge invariant, Wilson loop operators, $U_\Gamma = e^{-i\oint_\Gamma \vec{A}\cdot d\vec{\ell}}$, which measure the magnetic flux through a closed oriented loop Γ, where we use integral notation to indicate the product of $e^{-iA_{uv}}$ over all links (u, v) on the perimeter of Γ, with orientation along that of Γ. On planar graphs, which have trivial homology, an arbitrary Wilson loop can be decomposed into a product of small loop operators circling the elementary faces (plaquettes) of the graph, which we label by $\mathcal{F}$.

For dimensions $d > 2$, ordinary Maxwell electrodynamics emerges as the continuum and weak-coupling limit of the minimal gauge invariant Hamiltonian:

$$H_{\text{Maxwell}} = \frac{K}{2}\sum_{uv\in\mathcal{E}} E_{uv}^2 - \sum_{f\in\mathcal{F}}(U_f + U_f^\dagger) \tag{14}$$

where U_f denotes the Wilson loop encircling face f in the right-handed sense, and K is a coupling constant. The first term represents an electric field line tension, whereas the second gives an energy cost to magnetic flux (which produces quantum dynamics for electric fields). For $d = 2$, the lattice-QED systems is confined by monopole/instanton proliferation for any non-zero electric field line tension, $K > 0$.

3.2 QED-Mixer for Network flow problems

To obtain a flow-conserving mixer, one can nominally choose any gauge-invariant lattice-QED Hamiltonian, replacing electric field variables with flow variables. We introduce a separate electric-field "flavor" for each type of commodity indicated by a superscript parenthetical index: $E^{(i)}$ with $i = 1 \dots k$. In practice, we will choose our mixing Hamiltonian as the minimal Maxwell Hamiltonian, since it contains only the minimal elementary Wilson loops, thereby simplifying its implementation. Furthermore, we will set the electric field tension K to zero, since the goal of a mixer Hamiltonian is to produce unbiased quantum tunneling between different flow configurations. Significant efforts have been devoted to developing various schemes for "qubitization" and quantum simulation of lattice gauge theories [21]. We will remain largely agnostic about the specific implementation details, however, it is crucial to truncate the range of electric field values to lie between $-c(u, v) \leq E_{u,v} \leq c(u, v)$. To this end, we modify the electric field raising operator $e^{-iA_{uv}}$ to annihilate $|c(u, v)\rangle$, without altering its action on other states. We refer to the resulting Hamiltonian:

$$H_{\mathrm{M,QED}} = -\sum_{i=1}^{k} \sum_{f \in \mathcal{F}} (U_f^{(i)} + \mathrm{h.c.}) \qquad (15)$$

as the QED-mixer. We require that sufficiently many elementary faces/plaquettes $f \in \mathcal{F}$ are included to provide a complete basis of graph cycles, so that evolution under H_m can transfer any flow-configuration to any other flow configuration. This is easy to satisfy for planar graphs, one can readily verify that $O(|\mathcal{V}|)$ applications of H_m connect any any two flow configurations (see Appendix A). We note that the circuit complexity of implementing this mixing Hamiltonian grows length with number of minimal cycles.

3.2.1 Avoiding Isolated Loop Generation

As written, the QED-mixer does not allow any flow constraint violations. However, this mixer still suffers from a potential problem: it can crate isolated loops of circulating flow that do not connect to sources or sinks (see Figure. 3). These isolated loops satisfy all flow constraints, but do not correspond to a physically relevant solution. One option is to simply retain these isolated loops throughout the QAOA, and prune them from the final solutions via classical post-processing. A potential drawback is that is that isolated loops may incur unphysical costs, and on large graphs, each valid path can be dressed with exponentially many isolated loops, each of which could incur an unphysical cost penalty, masking the true cost of the "pruned" post-processed solution during the QAOA optimization. Throughout the remainder of this paper, we will restrict our attention to problems

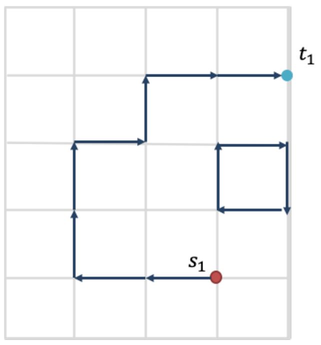


Figure 3: **A configuration with an isolated loop** Without the loop which is detached from the path, this would be a feasible solution. One could remove it easily, but having multiple isolated loops in a complicated graph would make such process hard to perform

with unit demand for each type of good. For this subclass of problems, we can avoid isolated loop creation by introducing further restrictions on the QED-mixer, which we call the restricted QED (RQED) mixer. In practice, this restriction will incur additional circuit complexity and may be undesirable. We will later compare the performance of the QED-mixer with and without restriction. The key step will be formulating a method to efficiently detect whether acting with U_f or $U_f^\dagger$ would create an isolated loop, depending on the graph property and specific problem. To avoid combinatorial blow-up of Hamiltonian terms, this detection must be done locally, which we do as follows. To determine whether adding electric field circulation around an elementary cycle of the graph adds an isolated loop, consider acting with $U_f^\dagger$ to add an electric field loop to a simple path and the following steps: Traverse the edge segments of the cycle in a counter-clockwise fashion. For each vertex $v \in \mathcal{V}$, count the number of electric field lines entering $(E_{v,\mathrm{in}}^{(i)})$ versus leaving $(E_{v,\mathrm{out}}^{(i)})$ the node. Denote their difference-squared as

$$\mathsf{V}^{(i)} \equiv \sum_{j=1}^{\ell} \left(E_{v_j,\mathrm{in}}^{(i)} - E_{v_j,\mathrm{out}}^{(i)} \right)^2, \qquad (16)$$

where $v_1, \dots, v_\ell$ are the nodes in the cycle. Notice that $(E_{v,\mathrm{in}}^{(i)} - E_{v,\mathrm{out}}^{(i)})^2$ can only take value 1 or 0. Since in our setting where maximum flow is 1, having two different direction of flows at the same node would suggest the node being used repeatedly, which further implies the configuration already contains an isolated loop. Imposing the Gauss' law constraint, $\mathsf{V}^{(i)}$ is equal to the total number of electric field lines entering or exiting the loop (if the loop does not contain a source/sink; or one could interpret a source as outside flow entering the loop and vice versa) without regard to sign (which is necessarily even). One can readily check that an isolated loop will be created unless $\mathsf{V}^{(i)} = 2$ (see Figure. 4 for sample instances).

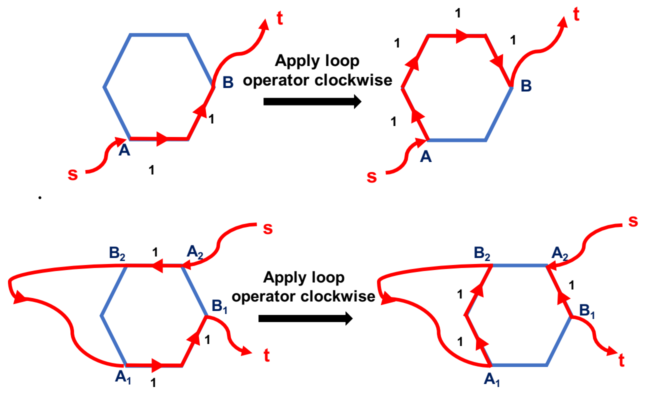


Figure 4: **An explanation of the "decision function"** For simplicity, we consider only one type of flow with max capacity 1. In both pictures, a flow (marked red) initially travels through the plaquette and then a loop operator is applied, increasing the flow on each edge on the plaquette by 1 in clockwise direction. The only difference between the pictures is that, the flow enters the loop twice at A_1 and A_2, and applying the operator resulted in redirected flow from A_1 to B_2, resulting in an isolated loop $A_1, B_2, ..., A_1$. To avoid such instances, we only apply the loop operator when exactly one continuous path of flow appears in the plaquette, which can be determined locally.

With this in mind, we can then left-multiply $U_f^\dagger$ by a locally evaluable "decision function" to define a modified loop operator:

$$U_f^{(i)} \to \tilde{U}_f^{(i)} = \delta_{\mathsf{V}^{(i)},2} U_f^{(i)}, \tag{17}$$

which does not create isolated loops. Note that $\delta_{\mathsf{V}^{(i)},2}$ commutes with U so the multiplication order is arbitrary.

In practice, $\delta_{\mathsf{V}^{(i)},2}$ can be written as a polynomial with zeros at all even values of $\mathsf{V}^{(i)}$ other than 2:

$$\delta_{\mathsf{V}^{(i)},2} = \prod_{j=0,1...\ell; j\neq 1} \left(\frac{2j - \mathsf{V}^{(i)}}{2j-2}\right), \tag{18}$$

which permits implementation with circuit complexity $\sim$ poly(ℓ). For simple graph structures, such as grids, where the sizes of elementary cycles are bounded independent of the system size, imposing this restriction adds only constant circuit-depth overhead.

3.2.2 Initial State Preparation

To begin the QAOA procedure, one must choose an initial state that is a quantum superposition with weights on all possible solutions. In the original formulation of QAOA, the initial state was chosen as the ground-state of the X-mixer Hamiltonian. This had two virtues: first, it ensured that QAOA could reduce to the quantum adiabatic algorithm in the limit of large step number, p. Second, this state is an equal weighted superposition of all computational states, and does not introduce an intrinsic bias.

In contrast, for QED-mixers, the mixer ground-state is no longer an equal-weight superposition. Moreover, it is not straightforward to implement the ground-state of the QED or RQED mixers. For these reasons, we consider alternative state preparation schemes. As a starting point, we assume that it is straightforward to greedily prepare a computational basis state that satisfies the flow-constraints (a detailed prescription will be given below for EDP problems).

Adiabatic ground-state preparation by reverse-annealing: One option would be to adiabatically prepare the QED or RQED mixer ground-state via adiabatic evolution from a classical Hamiltonian with the fixed computational basis state as the ground-state to the (R)QED mixer ground-state. However, generically, the QED mixer will have gapless photon-like excitations, whose gap scales to zero as $\sim 1/R$ where R is the graph radius (maximal distance between two nodes), such that this adiabatic ground-state preparation requires time $\sim \mathcal{O}(R)$. Moreover, we will see that starting from the ground-state of the mixer Hamiltonian actually leads to worse QAOA performance, due to the reasons we will discuss in later sections.

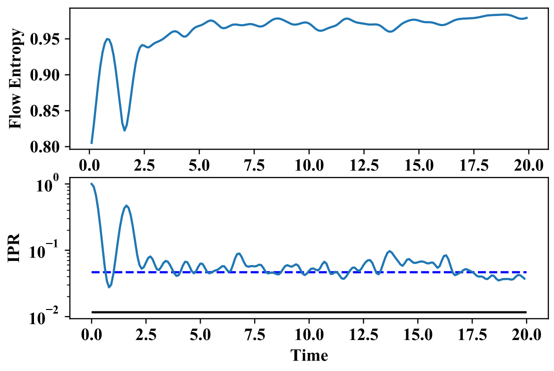
State preparation by mixer evolution: An alternative approach is to simply time-evolve the initial flow-constraint-preserving computational basis state with the mixing Hamiltonian for a certain amount of time, which spreads out the weight of the Hamiltonian onto other configurations. In analogy to photon propagation in electrodynamics, the flow should spread out ballistically (moving with constant velocity), covering the graph in time $\sim O(R)$. Hamiltonian simulation techniques can implement time-evolution for time t with performance that asymptotically tends to $O(t)$ [4]. In practice, it may not be necessary to simulate continuous time evolution, but rather one could break H_M into local terms acting on disjoint sets of qubits and stroboscopically alternate among them to achieve similar results.

To numerically analyze the spreading of the wave function, we introduce the inverse participation ratio (IPR) test:

$$\text{IPR} = \sum_i |\psi_i|^4, \tag{19}$$

where ψ_i is the amplitude of the wave-function in computational basis state i. IPR measure is inversely proportional to how evenly the wave-function spread-out over the computational basis states (i.e., among potential solutions to the optimization problem). The choice of power 4 here is because 2 would always give 1 and higher powers contain the same information about the wave function as 4th power does except for rare cases. When the wave-function is concentrated on a single state, IPR = 1; whereas an equal superposition of all states yields the minimal value of IPR $= 1/|\mathcal{H}|$, where $\mathcal{H}$ is the size of the Hilbert space (number of feasible solutions)

a)



b)

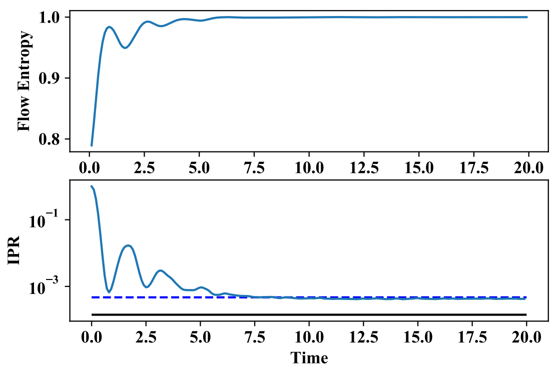


Figure 5: **IPR and entropy test** – The figures show instances of IPR and flow entropy (normalized to its maximum) S in the feasible space for single-source 4×4 (top) and 5×5 (bottom) square lattices for RQED-mixer in real time evolution. In IPR tests, the black solid line stands for the minimum possible IPR value (equal superposition of all possible paths from s to t), and the blue dashed line shows the IPR for mixer ground state. All $s - t$ pairs and initial paths are drawn at random. At both sizes, the entropy curve characters the bumps and saturation in the IPR curve, suggesting itself as a good alternative of IPR.

Figure. 5 shows the evolution of IPR with evolution under the RQED-mixer for single source-sink pairs on different sized square-grids. Since the RQED-Hamiltonian only evolves in the feasible solution space the test is done only within a constructed feasible subspace. The IPR decays from one, approximately saturating to a value close-to, but below the IPR for the mixer ground-state (blue dashed line), in characteristic time $t_{\text{sat}} \sim O(R)$ (where R is the graph radius). In addition, the IPR exhibits approximately periodic revival behaviors, which are most evident on the smaller 4×4 grid. As is well known from the study of Poincare recurrences, the period of these revivals becomes (doubly)-exponential in size of the graph, since the number of feasible solutions grows exponentially with the size of the graph, and can be safely neglected even for moderate graph sizes (indeed the oscillations are negligible already for the 5×5 grid.)

To prepare the initial state for subsequent QAOA iterations, we evolve the state until it just enters the saturation region where the IPR stabilizes to its long time value (e.g. in the 5×5-grid this occurs around $t_{\text{sat}} \approx 7.5$, see Figure. 5). In both tests, we observe that, inside the saturation region, the saturation-value of IPR lies below that of the mixer's ground-state, indicating that the mixer ground-state is more biased than the time-evolved state. This feature is natural since the evolved state is not low-energy and can be expected to contain additional configurational entropy.

In practice, IPR is challenging to measure as the Hilbert space size grows exponentially. Instead, one can determine the saturation time by monitoring local observables that act as witnesses for the IPR. Without loss of generality we focus on a single commodity case, since for multiple commodities, the Hilbert space is a tensor product of the single-commodity Hilbert spaces, with no inter-commodity interactions in the state preparation procedure. We examine the probability of observing unit flow (of either sign) on edge $e \in \mathcal{E}$ after evolution for time t under the mixing Hamiltonian

$$p_e(t) = \frac{\langle E_e^2 \rangle}{\sum_{e \in \mathcal{E}} \langle E_e^2 \rangle} \tag{20}$$

which can be estimated by sampling from the state in the computational basis.

We then define the (normalized) "flow entropy" as the von-Neumann entropy of this probability distribution:

$$S_f = -\frac{1}{|\mathcal{E}| \log 2} \sum_{e \in \mathcal{E}} p_e \log(p_e). \tag{21}$$

Larger $S_f \leq 1$ represents a more even distribution of paths. S_f saturates its maximal value of 1 when each link carries flow with equal probability. The flow entropy exhibits similar saturation behavior to the IPR, allowing one to measure the saturation time for a given graph. Crucially, to accurately estimate flow, the probability p_e needs only be measured to accuracy $\sim 1/|\mathcal{E}|$, which requires sampling cost $\sim |\mathcal{E}|^2$ that is polynomial in problem size (in contrast to the exponentially small IPR), allowing an efficient measurement to identify saturation time at which to stop the state preparation step.

3.3 Algorithm description

We are now ready to detail the steps of the modified QAOA for network flow problems. Given a directed graph $G(\mathcal{V}, \mathcal{E})$ as input (if the graph is undirected, simply choose an arbitrary orientation for the edges):

1. *Pre-process:* Identify a set of elementary faces (i.e. choose a basis of closed cycles) in G and store them. For a planar graph, this can be done classically in polynomial time [7].

2. *Hamiltonians simulation:* Choose a technique to simulate time-evolution under the cost and mixing Hamiltonians: H_C, H_M.

3. *Initial state preparation:* As described in Section 3.2.2, for each pair (s_i, t_i) given in the input, pick an arbitrary "seed" path, P_0 (which can be found efficiently by standard methods), and define the corresponding computational basis state as $|P_0\rangle$. Identify the saturation time t_{sat}, for the graph by the flow-entropy test described in the text. Then, simulate time-evolution under the mixing Hamiltonian to form the initial state: $|\psi_0\rangle = e^{-iH_M t_{\text{sat}}}|P_0\rangle$.

4. *Variational Optimization:* Following the original QAOA procedure, but replacing the the X-mixer with the (R)QED-mixer to avoid generated flow-constraint violations, find $\boldsymbol{\gamma}_*, \boldsymbol{\beta}_* = \arg\min \varepsilon_C$ using any desired classical minimization procedure,

5. *Post-process* Repeatedly sample from the optimized variational state $|\psi(\boldsymbol{\gamma}_*, \boldsymbol{\beta}_*)\rangle$, recording the best (lowest-cost) sample encountered as an approximate solution.

4 Numerical simulation of algorithm performance

In this section we present results from numerical simulation of QED-modified and standard QAOA of small-scale network flow problems. Due to the rapid growth of Hilbert space, $|\mathcal{H}| \sim O(3^{k|\mathcal{E}|})$, the accessible problem size is quite limited. In order to provide a meaningful comparison of the QED-mixer, we first consider the (classically easy) SSSP problem ($k = 1$), which will allow simulation of relatively larger graphs to enable a comparison of QED-mixer and X-mixer. We then simulate EDP problems with $k = 2$ on a grid graph for RQED-mixer only, where we can restrict our numerical simulation to the feasible solution space of size $\ll |\mathcal{H}|$.

For the original X-mixer, in each step, the variational parameters can be limited to $[0, 2\pi]$ for $\{\gamma_i\}$ and $[0, \pi]$ for $\{\beta_i\}$, due to the periodicity of evolution under Pauli strings. The QED-mixer has no such periodicity. However, to run the QED-mixer for longer times would require additional circuit depth with which additional rounds of QAOA with the X-mixer could have been performed. Hence, to make a fair comparison, we also restrict our variational parameter ranges for the QED-mixer to the same range as for the original X-mixer.

In all simulations, we first perform a global search with differential evolution, and then optimize with a local BFGS method [15]. For both methods, we restrict the optimizer to at most 200 minimization steps to balance accuracy and efficiency.

To generate a larger collection of problems from a limited set of graph types and sizes, we generate random problem instances for each graph. For the SSSP problems we consider for each triangle graph in Figure. 2 with source-and-sink located at opposite corners, we generate random problem instances by drawing random weights w_e i.i.d. for each edge from the uniform distribution on the unit interval $[0, 1]$, and seeding the state-preparation step with a uniformly-randomly chosen path, $|P_0\rangle$. For the EDP problem, the edges are unweighted, so we further choose the source and sink locations uniformly at random on different sized grid graphs.

4.1 Comparing Mixers

To compare the performance of QAOA on network flow problems using different X-, QED-, and RQED-mixers, we adopt a metric called the approximation ratio (AR) [23], defined as:

$$\text{AR}(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \frac{\langle \psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta})|\Pi\,(C_{\max} - H_C)\,\Pi|\psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle}{C_{\max} - C_{\min}} \tag{22}$$

where $C_{\max}$ and $C_{\min}$ respectively represent the maximum and minimum costs from the set of feasible solutions, and Π is the projector into the feasible subspace, which ensures that only states without constraint violations and isolated loops are counted. The approximation ratio indicates fractional of improvement compared to the worst case, normalized by the possible range of cost values, despite whether an EDP instance on a certain problem exists. In practice, we perform multiple independent runs to obtain average performance, namely, the average approximation ratio (AAR), as the indicator of QAOA performance. Similarly, the variational optimization of QAOA parameters is done with respect to the projected cost function:

$$\tilde{\varepsilon}_C(\boldsymbol{\gamma}, \boldsymbol{\beta}) := \langle \psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta})|\Pi H_C \Pi|\psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle. \tag{23}$$

Whereas, by construction, the QED- and RQED-mixers automatically avoid flow-conservation violating constraints, flow-constraint violations can only be softly penalized by introducing an extra term to the cost function for the X-mixer:

$$H_{C,\text{penalty}} = \Delta \sum_{u \in \mathcal{V}, i} \left(\sum_{(u,v) \in \mathcal{E}} E^{(i)}_{(u,v)} - d_i(\delta_{u,s_i} - \delta_{u,t_i}) \right)^2. \tag{24}$$

In principle, Δ introduces an extra hyperparameter that must be optimized. Generally, Δ should increase

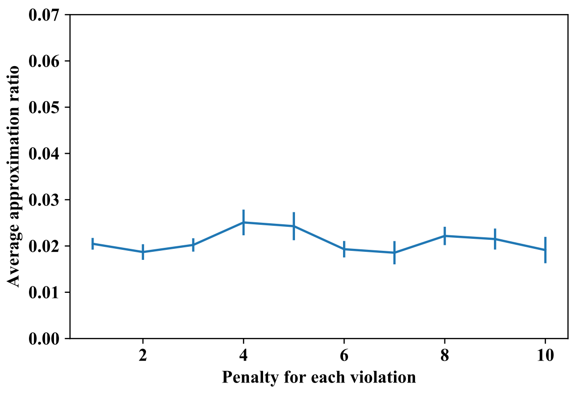


Figure 6: **Behavior of X-mixer QAOA with different penalties,** Δ for SSSP problem at $P = 1$. This result shows that the average behavior of X-mixer QAOA is fairly insensitive to the precise choice of the penalty coefficient Δ.

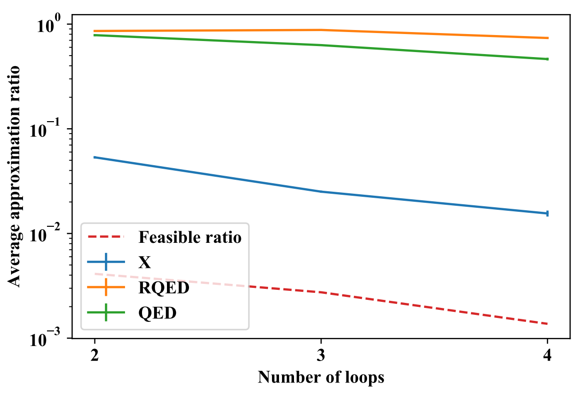
with problem size to avoid the tendency to lower cost by violating constraints. For the problem-sizes we simulate, the results are not very sensitive to the precise choice in Δ (Figure. 6), and we choose $\Delta = 1$ throughout for simplicity.

4.2 Mixer comparison on SSSP problems

We begin with a comparison of the performance between all three mixers: the X-, QED- and RQED-mixer, for approximately solving SSSP problems on different sized graphs. As expected, X-mixer exhibits substantially worse performance than the flow-constraint preserving QED mixers. For a single QAOA round, $p = 1$, the degradation in X-mixer's performance with increasing graph sizes tracks the decreasing trend of the ratio between the number of feasible solutions and the size of whole Hilbert space (as shown in Figure. 7).

The unrestricted QED mixer initially matches the RQED-mixer on the smallest problem instances, for which the graphs are too small to permit isolated loop creation. As the graph size grows, the unrestricted QED mixer's AAR drops below that of the RQED-mixer. For the largest graphs, the QED-mixers' AAR approaches the value achieved by picking feasible paths at random, showing that isolated loop creation can substantially degrade the unrestricted QED-mixer performance at $p = 1$.

This shows that, although we start with a feasible solution, isolated loops can be created when using the QED-mixer in its original version. A multistep QAOA simulation shows that, for the 2-triangle graph, the QED-mixers are able to solve the problem exactly at around $p = 3$, which is not surprising due to the small size of the problem.



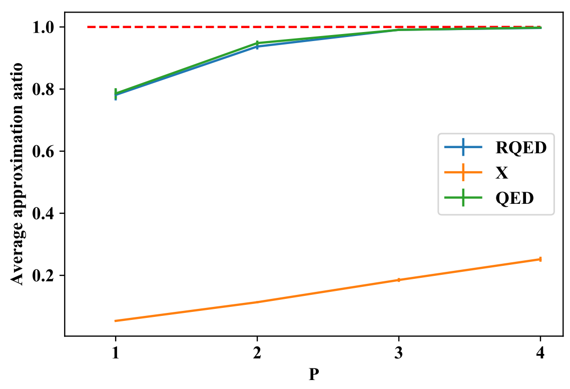


Figure 7: **Comparing different mixers Top:** Solving SSSP on different sized triangle graphs with different mixers; 120 runs performed for each mixer: weight on each edge is randomly drew from $[0, 1]$. **Bottom:** A multiple-step comparison: We compare the behavior of the 3 mixers in solving SSSP problem on the 2-triangle graph as plotted in Figure 2. Each point represents an average of 200 random instances. Notice that, for this particular graph, it is impossible for the unrestricted QED-mixer to create an isolated loop, making its performance almost identical to the RQED-mixer.

4.3 EDP on Undirected Graphs

Even though a direct comparison between the X-mixer and the QED-mixers for EDP problems is expensive, we test out the performance of the QED-mixers alone on larger graphs by restricting the simulation to the feasible subspace to reduce the computational power required. In order to be able to compare performances at different graph sizes, we only consider EDP problems with $k = 2$ source-sink pairs. As shown in Figure. 8, even though the solution space size for 4×4 grid is typically 100 or more times (depending on the location of sinks and sources) than that of the 3×3-grid, the performance is only weakly affected – even after only a single QAOA round, $p = 1$, the AAR remains higher than 0.7. As a complementary to the results in IPR test, Figure. 9, shows how different initial state IPRs result in different outcomes in solving EDP on 3×3 grids. We observe that, the ground state preparation is not a necessity for our mixer, but

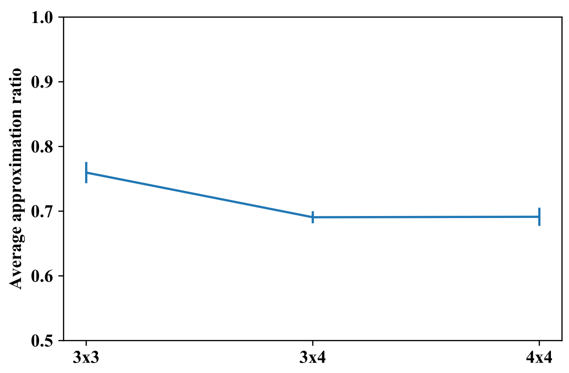


Figure 8: **RQED-Mixer Behavior at $p = 1$** The simulation is done for 3×3, 3×4, 4×4 grids for a 2-pair EDP problem. 200 random problem instances are performed at each graph, by choosing the location of each source, sink, state preparation seed path at random.

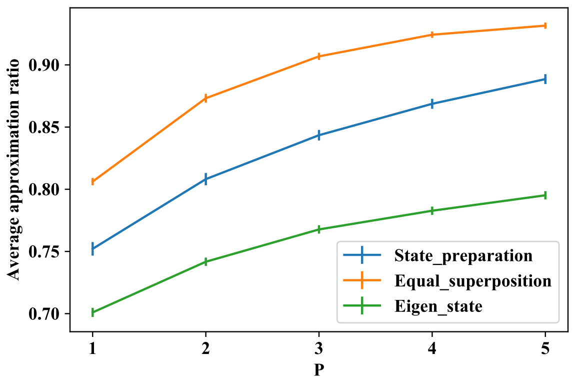


Figure 9: **RQED QAOA behavior in solving actual EDP problems with different initial states** We compare the effect of different choices of initial states on the RQED-mixer's performance, averaging over 200 random problem instances. "Eigenstate" stands for the ground state.

the equal superposition state of all feasible solutions does serve as a best starting point of the three, followed closely by the initial state prepared by evolving a random configuration with the mixing Hamiltonian, which is with the IPR test.

These results suggest that having an unbiased ergodic superposition of solutions is more advantageous than starting close to the mixer ground-state (for ordinary QAOA with the X-mixer, these coincide).

5 Discussion

In this work, we designed and simulated a QED-inspired QAOA algorithm to the flow network problems. In particular, we tested its performance with the EDP and SSSP problems. The biggest difference between routing problems and other typical QAOA benchmark problems (like MaxCut) is that the feasible solutions only consist of an exponentially small fraction of the whole solution space. The standard QAOA approach produces infeasible solutions with high probability. To resolve this issue, we proposed the RQED-mixer, which automatically ensures the satisfaction of flow constraints throughout the algorithm. By observing the analogy between Gauss' Law and those constraints, we theoretically and numerically demonstrated that the QED-mixer is a natural choice for the routing problem. Although implementing the RQED-mixer requires additional circuit complexity, the generating Hamiltonian is still local and the number of terms is still linear in problem size, and optimization purely within the feasible space makes the QAOA with RQED-mixer more likely to find nearly optimal solutions in comparison to the standard QAOA approach. Part of the simulation results showed that for SSSP problem, the average approximation ratio of RQED-mixer is significantly higher than the X-mixer. For the harder problem, EDP, our results also showed that QAOA with RQED-mixer can achieve high approximation ratio on different size instances, although our numerical simulations were necessarily limited to rather modest problem sizes.

Our experiments with different initial state strategies suggest an intriguing departure from the "shortcut-to-adiabaticity" mechanism typically used to motivate QAOA. Namely, QAOA is often motivated as a short-depth approximation to the adiabatic mapping from mixer to the cost of ground-state. However, we have seen that, at least on modest graph sizes available for classical simulation, starting with a more ergodic (less biased) superposition of initial states produces better results than starting in a low-energy state of the mixer, suggesting that a different mechanism than approximate adiabaticity is at play.

Whether the improved performance and superiority of the non-adiabatic operations extend to a larger problem size is an important question for future studies. However, the scope of classical simulation is limited due to the typical explosion of Hilbert space size with problem size. Analytic insights would be extremely valuable, although have often proved challenging beyond small-p. One possible approach is to investigate the locality of QAOA with the RQED-mixer. For standard QAOA with X-mixer, the locality was studied [13] to prove the performance of QAOA on the independent set problem, another famous NP-complete problem on graph. Last but not least, it would be desirable to implement the algorithm on near-term quantum computers, as these devices begin to eclipse classical simulation [1].

Acknowledgements

We thank David Hayes for insightful conversations. The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing high-performance computing resources that have contributed to the simulation results reported within this paper. URL: http://www.tacc.utexas.edu. This work was supported by NSF DMR-1653007 (AP) and NSF Grant CCF1648712 (RZ).

A Proof of the QED-mixer's universality for planar graphs

In this section we prove that QED-mixer is able to evolve between two arbitrary paths in $O(n)$ loop operations on a planar graph. *Given a undirected graph $G(V, E)$ with k pairs (s_i, t_i). The goal is to find k paths connecting s_i and t_i for all $i \in [k]$ such that the maximum of congestion in each edge is minimized.*

Proof: We first assume that P_1 and P_2 do not have any common vertex. Then, $s \xrightarrow{P_1} t \xrightarrow{-P_2} s$ forms a closed simple region, where $-P_2$ means the inverse direction of path P_2. By Jordan's theorem [18], we can take the "interior" of this region, which is a subgraph G' of G. It's easy to see that every cycle in G' is also a cycle in G. Hence, we can apply a loop operation for every cycle in G', and doing so in some specific directions will transform P_1 to P_2. Wlog., suppose $s \xrightarrow{P_1} t \xrightarrow{-P_2} s$ is in clockwise direction. Then, for every cycle, we apply a counter-clock loop operator, which is equivalent to transfer 1 unit of flow counterclockwise through the cycle. Let $e = (u, v)$ be an edge in G'. If e is contained in P_1 and initially there is 1 unit flow from u to v. After the loop operation, another 1 unit flow from v to u is introduced so that the total flow on e is 0. Similarly, if e is contained in P_2 and is in the same direction as P_2, then the flow on e is 1. For all the interior edges, the flow on them is 0 because each edge is contained in two cycles and the loop operation on each cycle will introduce 1 unit flow through e in opposite directions, which will be cancelled by each other. Therefore, after these loop operations, the flow from s to t through P_1 will be transformed to P_2.

In general, let $v_1 = s, \ldots, v_l = t$ be l common vertices between P_1 and P_2, sorted by their appearance orders in the path. Then, we can see that for all $i \in [t-1]$, $v_i \to v_{i+1} \to v_i$ forms a closed simple region and we take the subgraph G'_i. For each G'_i, we can apply a series of loop operations to transform $v_i \xrightarrow{P_1} v_{i+1}$ to $v_i \xrightarrow{P_2} v_{i+1}$. Therefore, after processing $t-1$ subgraphs, P_1 will be transformed to P_2.

Lastly, we show that the number of loop operations we applied is $O(n)$. For each cycle, we only apply the corresponding loop operation once. Hence, the number of loop operations is upper bounded by the number of cycles in G. Since G is planar, Euler characteristic for planar graph gives $n - m + f = 2$, where m is the number of edges in G and f is the number of cycles. Thus, we have $f = m - n + 2$. We also know that, for planar graph, $m \leq 3n - 6$. Hence, $f \leq 2n - 4 = O(n)$. Therefore, we can transform P_1 to P_2 by $O(n)$ loop operations.

B Dual "height-model" formulation

Here we consider a canonical (for a detailed description, see, [14], for example) dual picture description of the algorithm on plane graphs that might be useful for implementation sometimes. In graph theory, the dual of any plane graph G is obtained by taking each of its faces as a vertex, and drawing an edge between any two neighboring faces. In this dual representation, an initial configuration is chosen, and the states are defined by the relative "loop distance" to the initial configuration. The amount of flow on each path is then equal to the initial flow plus the difference between states of adjacent faces with the direction perpendicular counterclockwise to the gradient direction. Namely, a "1" state on some elementary loop adds a counterclockwise flow loop to the initial configuration, and vice versa. Naively, one would think that

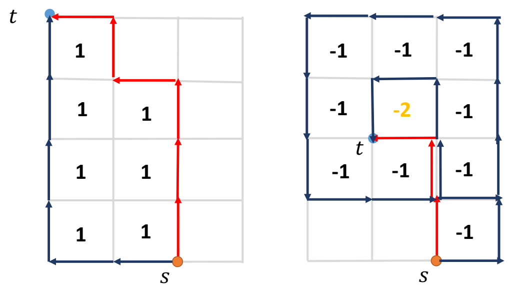


Figure 10: **Examples of dual picture description** We consider a single-pair flow instance: red represents initial/reference configuration, and blue stands for the final configuration. Numbers on each faces stands for the states encoded in dual picture language; every unlabeled face has state 0. In some cases (left) the range of dual state could be simply $-1, 0, 1$ whereas more complicated paths (right) needs greater range, which could be proportional to the radius of the graph, namely $O(\sqrt{n})$.

the total Hilbert size for EDP problem becomes 3^{kf}, where f stands for the number of faces; this makes the Hilbert space a polynomial order less than the original picture, considering $e > f$ on planar graphs. In addition, we could prepare equal superposition states in the dual picture. Nevertheless, there are cases which require more levels on each face, as shown in Figure. 10. For larger graphs, the dual encoding thus becomes even more expensive. On the other hand, the encoding still cannot get rid of isolated loops, although a direct interpretation of RQED-mixer is possible. In conclusion, the dual description can be a useful alternative when considering the ordinary QED-mixer on small graphs, but adds significant qubit resource overheads for larger graphs.

References

[1] Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505–510, 2019. DOI: https://doi.org/10.1038/s41586-019-1666-5.

[2] Boaz Barak, Ankur Moitra, Ryan O'Donnell, Prasad Raghavendra, Oded Regev, David Steurer, Luca Trevisan, Aravindan Vijayaraghavan, David Witmer, and John Wright. Beating the random assignment on constraint satisfaction problems of bounded degree. *CoRR*, abs/1505.03424, 2015. URL `http://arxiv.org/abs/1505.03424`.

[3] Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

[4] Dominic W. Berry, Andrew M. Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, Oct 2015. DOI: https://doi.org/10.1109/focs.2015.54. URL `http://dx.doi.org/10.1109/FOCS.2015.54`.

[5] Sergey Bravyi, Alexander Kliesch, Robert Koenig, and Eugene Tang. Obstacles to variational quantum optimization from symmetry protection, Dec 2020. URL `https://doi.org/10.1103/PhysRevLett.125.260505`.

[6] Julia Chuzhoy and Shi Li. A polylogarithmic approximation algorithm for edge-disjoint paths with congestion 2. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 233–242, 2012. DOI: 10.1109/FOCS.2012.54.

[7] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008. ISBN 9783540779735. URL `http://www.worldcat.org/oclc/227584184`.

[8] Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[9] Shimon Even, Alon Itai, and Adi Shamir. "on the complexity of time table and multi-commodity flow problems". In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pages 184–193. IEEE, 1975.

[10] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm. *arXiv e-prints*, art. arXiv:1411.4028, November 2014. URL `https://ui.adsabs.harvard.edu/abs/2014arXiv1411.4028F`.

[11] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem, 2014. URL `https://arxiv.org/abs/1412.6062`.

[12] Edward Farhi, David Gamarnik, and Sam Gutmann. The quantum approximate optimization algorithm needs to see the whole graph: Worst case examples, 2020. URL `https://arxiv.org/abs/2005.08747`.

[13] Edward Farhi, David Gamarnik, and Sam Gutmann. The quantum approximate optimization algorithm needs to see the whole graph: A typical case, 2020. URL `https://arxiv.org/abs/2004.09002`.

[14] Matthew P. A. Fisher. Chapter 13 duality in low dimensional quantum field theories. 2018. DOI: https://doi.org/10.1007/978-1-4020-3463-3˙13.

[15] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[16] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917855. DOI: 10.1145/237814.237866.

[17] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor G Rieffel, Davide Venturelli, and Rupak Biswas. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *Algorithms*, 12(2):34, 2019. DOI: 10.3390/a12020034.

[18] Thomas C Hales. The jordan curve theorem, formally and informally. *The American Mathematical Monthly*, 114(10):882–894, 2007. DOI: https://doi.org/10.1080/00029890.2007.11920481.

[19] Matthew B Hastings. Classical and quantum bounded depth approximation algorithms. *arXiv preprint arXiv:1905.07047*, 2019. DOI: https://doi.org/10.26421/QIC19.13-14-3.

[20] KR Khadiev and LI Safina. Quantum algorithm for shortest path search in directed acyclic graph. *Moscow University Computational Mathematics and Cybernetics*, 43(1):47–51, 2019. DOI: https://doi.org/10.3103/S0278641919010023.

[21] D. Marcos, P. Widmer, E. Rico, M. Hafezi, P. Rabl, U.-J. Wiese, and P. Zoller. Two-dimensional lattice gauge theories with superconducting quantum circuits. *Annals of Physics*, 351:634–654, Dec 2014. ISSN 0003-4916. DOI: 10.1016/j.aop.2014.09.011.

[22] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994. DOI: 10.1109/SFCS.1994.365700.

[23] Zhihui Wang, Nicholas C Rubin, Jason M Dominy, and Eleanor G Rieffel. *xy* mixers: Analytical and numerical results for the quantum alternating operator ansatz. *Physical Review A*, 101(1):012320, 2020. DOI: 10.1103/PhysRevA.101.012320.