arXiv:2406.14330v4 [quant-ph] 30 Jul 2026

Promise of Graph Sparsification and Decomposition for Noise Reduction in QAOA: Analysis for Trapped-Ion Compilations

Jai Moondra[1], Phillip C. Lotshaw[2], Greg Mohler[3], and Swati Gupta[4]

[1]Carnegie Mellon University
[2]Oak Ridge National Laboratory
[3]Georgia Tech Research Institute
[4]Massachusetts Institute of Technology

We develop new approximate compilation schemes that significantly reduce the expense of compiling the Quantum Approximate Optimization Algorithm (QAOA) for solving the Max-Cut problem. Our main focus is on compilation with trapped-ion simulators using Pauli-X operations and all-to-all Ising Hamiltonian H_{Ising} evolution generated by Mølmer-Sørensen or optical dipole force interactions, though some of our results also apply to standard gate-based compilations. Our results are based on principles of graph sparsification and decomposition; the former reduces the number of edges in a graph while maintaining its cut structure, while the latter breaks a weighted graph into a small number of unweighted graphs. Though these techniques have been used as heuristics in various hybrid quantum algorithms, there have been no guarantees on their performance, to the best of our knowledge. This work provides the first provable guarantees using sparsification and decomposition to improve quantum noise resilience and reduce quantum circuit complexity.

For quantum hardware that uses edge-by-edge QAOA compilations, sparsification leads to a direct reduction in circuit complexity. For trapped-ion quantum simulators implementing all-to-all H_{Ising} pulses, we show that for a $(1-\epsilon)$ factor loss in the Max-Cut approximation ($\epsilon > 0$), our compilations improve the (worst-case) number of H_{Ising} pulses from $O(n^2)$ to $O(n\log(n/\epsilon))$ and the (worst-case) number of Pauli-X bit flips from $O(n^2)$ to $O\left(\frac{n\log(n/\epsilon)}{\epsilon^2}\right)$ for n-node graphs. This is an asymptotic improvement for any constant $\epsilon > 0$. We demonstrate that significant improvements to the approximation ratio are obtained using decomposition in simulated trapped-ion experiments with dephasing noise. We further present a generic argument showing that sparsification results in an exponentially improved circuit fidelity lower bound in digital computing schemes based on one- and two-qubit gates, which are relevant to a wide variety of hardwares such as superconducting qubits and certain neutral atom or trapped ion setups, and more sophisticated noise models. We anticipate these approximate compilation techniques will be useful tools in a variety of future quantum computing experiments.

Keywords: QAOA, Max-Cut, NISQ computing, graph sparsification[1]

Jai Moondra: jmoondra@andrew.cmu.edu, corresponding author. Work done while the author was at Georgia Institute of Technology.

[1]This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

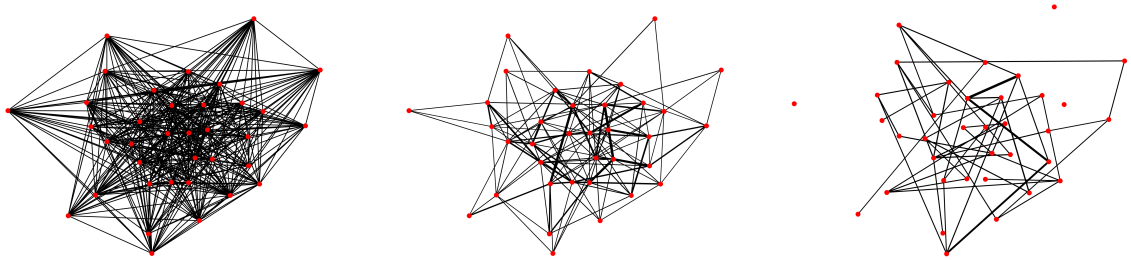


Figure 1: Left to right: graph sparsification, with the original unweighted graph G (left, 397 edges) and two weighted graph sparsifiers H_1 (middle, 136 edges) and H_2 (right, 48 edges) of G (all graphs have the same number of vertices). The Max-Cut in H_1 gives a cut in G with cut value that is 90% of the Max-Cut in G, i.e., is a 0.9-approximation to Max-Cut in G. The Max-Cut in H_2 is a 0.82-approximation to Max-Cut in G.

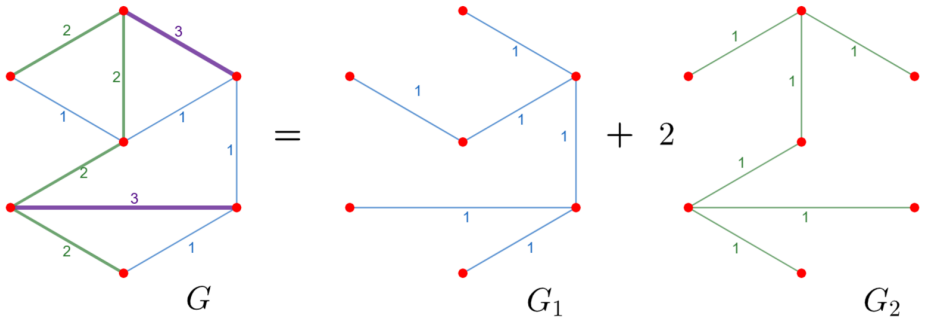


Figure 2: The graph decomposition process: The weighted graph G on the left can be written as the weighted sum $G_1 + 2G_2$ of two unweighted graphs G_1, G_2 on the right.

1 Introduction

The advantage of quantum computers over their classical counterparts for some computational tasks has been established for decades [56, 22]. Promising new quantum algorithms for complex problems such as constrained optimization [27, 2, 42] and combinatorial optimization [18, 46, 60, 59, 55, 23, 17, 19, 41] are an active area of research. A leading benchmark for quantum optimization is the Max-Cut problem: given a graph $G = (V, E, c)$ on vertices V, edges E, and edge weights $c : E \to \mathbb{R}$, one seeks to find the subset $S \subseteq V$ maximizing the weighted cut value $\delta(S) := \sum_{uv \in E : u \in S, v \in V \setminus S} c_{uv}$. The development of the Quantum Approximate Optimization Algorithm (QAOA) [18] for Max-Cut has made it a leading candidate for demonstrating quantum advantage over classical computing.

While classical algorithms currently have much faster runtimes, they are widely believed to have fundamental limitations when it comes to the solution quality, which is measured by the approximation ratio, defined as the ratio of the algorithm's cut value to the optimal cut value [64, 67]. The state-of-the-art for Max-Cut is the Goemans-Williamson (GW) [21] algorithm with (tight [26]) approximation ratio $\alpha^* \simeq 0.878$ for all graphs with nonnegative edge weights. On the other hand, algorithms like QAOA are known to converge to the Max-Cut under the adiabatic limit (i.e., as the depth goes to infinity) [18]. Though known approximation bounds at finite circuit depths of QAOA do not outperform Goemans-Williamson's 0.878, computational advantages of QAOA relative to classical algorithms have been presented in varying contexts [61, 3, 19, 53, 69].

A significant bottleneck that hinders the attainment of quantum advantage with existing hardware is the loss in performance due to quantum noise arising from imperfect control or unwanted interactions with the environment. Limiting the impact of noise is therefore important both for progress [51, 11] towards building fault-tolerant quantum computers [56, 57, 29] and for solving medium to large problem

instances with near-term Noisy Intermediate-Scale Quantum (NISQ) computers [49, 13]. There has also been interest in using special-purpose quantum simulators with global addressing for quantum optimization at scales that are currently intractable with local one- and two-qubit gate addressing [50, 15, 45].

In this work, we approach quantum noise reduction from a *problem-aware* perspective, and ask:

Is it possible to classically modify the problem instance to improve quantum noise resilience while still being able to recover the correct solution? Are there provable guarantees for such techniques?

In the context of Max-Cut and QAOA, we answer these questions affirmatively using two classical pre-processing steps to reduce circuit complexity: graph sparsification and graph decomposition. Graph sparsification is a principled approach to reduce the number of edges $m = |E|$ in the graph and introduce weights on remaining edges, while ensuring that every cut in the original graph is approximated by the corresponding cut in the sparsified instance. A higher level of graph sparsification degrades the approximation guarantee; see the example in Fig. 1. Sparsification has been used extensively in classical computing; however, its potential has not been well explored in quantum optimization. Recent work has used sparsifying heuristics to improve QAOA performance, but without theoretical guarantees [30].

Graph decomposition expresses a weighted graph as a weighted sum of a logarithmic (in the number of vertices, denoted $n = |V|$) number of unweighted graphs with controlled approximation error, as seen in Fig. 2. Decomposition is tailored to a specific compilation technique [50] with trapped-ion global gates, while sparsification is generally applicable to any QAOA implementation and may be especially beneficial for limiting SWAP gates in hardware with limited connectivity [33]. Both of these techniques provide more efficient compilations – requiring less time and fewer steps for execution – with less accumulated noise.

QAOA begins with a classical problem defined in terms of a cost function $\mathcal{C}(\boldsymbol{z})$ with a bit string argument $\boldsymbol{z} = (z_1, \dots z_N)$. This is mapped to a quantum *cost Hamiltonian* C with an eigenspectrum that contains the set of classical solutions $C|\boldsymbol{z}\rangle = \mathcal{C}(\boldsymbol{z})|\boldsymbol{z}\rangle$, such that identifying the ground state of C provides the optimal solution [37]. To approximately solve these problems, QAOA evolves a quantum state in p layers of Hamiltonian evolution, where each layer alternates between evolution under C and under $B = \sum_i \sigma_i^x$ where σ_i^x is the Pauli-X operator on qubit i,

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = \prod_{l=1}^{p} e^{-i\beta_l B} e^{-i\gamma_l C} |\psi_0\rangle, \tag{1}$$

where $|\psi_0\rangle$ is the initial state (i.e., $|+\rangle^n$ for standard QAOA, or a classically computed warm-start [16, 61]). The $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are variational parameters chosen to minimize $\langle C\rangle := \langle \boldsymbol{z}|C|\boldsymbol{z}\rangle$, such that measurements of the quantum state $|\boldsymbol{z}\rangle$ provide approximate solutions to the combinatorial problem.[2] Given graph $G = (V, E, c)$ with edge costs c_e, the expected sum of costs of cut edges across a cut $\delta_G(S)$ (for $S \subseteq V$), is related to the graph coupling operator expectation as $\delta_G(S) = \frac{1}{2}\left(\sum_{e\in E} c_e - \langle \boldsymbol{z}_S|C|\boldsymbol{z}_S\rangle\right)$, where $\boldsymbol{z}_S$ is the corresponding bit string. Therefore, minimizing $\langle C\rangle$ is equivalent to finding $S \subseteq V$ with the maximum cut value $\delta_G(S)$. We discuss this further in Section 2. In quantum computers with local-only addressing, each edge requires a noisy two-qubit gate operation [33] to compile $\exp(-i\gamma C)$. This case presents an obvious advantage for sparsified instances with fewer edges. For global-gate compilations more involved compilation schemes are required.

We focus on compilations for trapped-ion hardware with global Mølmer-Sørenson (MS) [31] or optical dipole force [8] interactions. These gates natively implement an equally-weighted all-to-all

[2]Note we have chosen a convention in which we are beginning in the ground state of $-B$ and aiming to prepare the ground state of $+C$, or equivalently beginning in the highest state of $+B$ and trying to prepare the highest state of $-C$. The initial and target states are adiabatically connected in the limit $p \to \infty$ for the usual reasons [18]; see also Ref. [5] for a rigorous proof of convergence criteria under different choices of B and C. We did not include the alternating signs directly in (1) to better match the standard conventions, instead absorbing them into the parameters, which have arbitrary signs.

Table 1: Worst-case bounds on number of H_{Ising} pulses (denoted $N(H_{\text{ISING}})$) and total number of operations (number of H_{Ising} pulses and bit flips, denoted $N(\text{TOTAL}_{\text{OPS}})$) on weighted and unweighted (dense) graphs on $|V| = n$ vertices and $|E| = m = O(n^2)$ edges. For any constant ϵ, our algorithms yield an asymptotic improvement in the number of pulses and total number of operations, while keeping 'large' or 'non-trivial' cut values (see Section 2 for formal definition) within a factor $1 \pm \epsilon$. The best bounds (up to log factors) for weighted graphs are highlighted in yellow; note that using sparsified + decomposed graphs achieves the best bounds for both $N(H_{\text{ISING}})$ and $N(\text{TOTAL}_{\text{OPS}})$ simultaneously.

Instance	Weighted?	Number of edges	Loss in cut approximation	$N(H_{\text{ISING}})$	$N(\text{TOTAL}_{\text{OPS}})$	Notes
Original Graph	×	$O(n^2)$	Exact	$O(n)$	$O(n^2)$	[50]
	✓	$O(n^2)$	Exact	$O(n^2)$	$O(n^2)$	
Decomposed Graph	✓	$O(n^2)$	$1-\epsilon$	$O\left(n \log \frac{n}{\epsilon}\right)$	$O(n^2)$	Theorem 4
Sparsified Graph	✓	$O\left(n/\epsilon^2\right)$	$1-\epsilon$	$O\left(n/\epsilon^2\right)$	$O\left(n/\epsilon^2\right)$	Follows from [4] and [50]
Sparsified + Decomposed Graph	✓	$O\left(n/\epsilon^2\right)$	$1-\epsilon$	$O\left(n \log \frac{n}{\epsilon}\right)$	$O\left(\frac{n \log(n/\epsilon)}{\epsilon^2}\right)$	Theorem 5

Ising evolution with Hamiltonian

$$H_{\text{Ising}} = \sum_{i<j} \sigma_i^z \sigma_j^z. \tag{2}$$

[50] show that arbitrary graph couplings for QAOA can be compiled on trapped-ion hardware using a sequence of global H_{Ising} pulses and individual "bit flips" (single-qubit Pauli-X unitaries) to produce effective Ising interactions localized to star graphs, which can be summed to generate C for any arbitrary graph. Their algorithm – called UNION-OF-STARS – compiles unweighted graphs with $O(n)$ H_{Ising} pulses while weighted graphs require $O(m) = O(n^2)$ H_{Ising} pulses in the worst-case; for weighted and unweighted graphs the number of bit flip operations is $O(m)$. Compilations that use fewer H_{Ising} pulses and bit flips can potentially reduce noise in the compiled circuit.

In our first contribution, we show that if a $1-\epsilon$ factor approximation loss in Max-Cut approximation can be tolerated, then a weighted graph G on n vertices can be written as a (weighted) sum of at most $O(\log(n/\epsilon))$ unweighted graphs using our decomposition technique. Here, $\epsilon > 0$ is an arbitrary number that can be chosen to suit the needs of optimization. The smaller ϵ is, the better the Max-Cut approximation. Consequently, we can compile G by composing the compilations of each of these unweighted graphs. This uses a total of $O(n \log(n/\epsilon))$ H_{Ising} pulses in the worst-case, which is asymptotically much smaller than the $O(m) = O(n^2)$ pulses needed by the edge-by-edge compilation in UNION-OF-STARS(e.g., $O(n \log n)$ vs $O(n^2)$ even for $\epsilon = 1/\log n$).

Next, we use classical graph sparsifiers to reduce the number of edges from m to $O\left(\frac{n}{\epsilon^2}\right)$. Our sparsification approach is applicable to quantum hardware with local or global addressing; for global gate compilations, applying graph decomposition to the sparsified graph yields compilations that use at most $O\left(\frac{n \log(n/\epsilon)}{\epsilon^2}\right)$ bit flips, again asymptotically smaller than the $O(m) = O(n^2)$ bit flips used by UNION-OF-STARS. These bounds are summarized in Table 1. We verify them through simulations on graphs from the MQLib library [14], which is a suite of graphs that serve as a benchmark for Max-Cut algorithms. *We observe up to* 80% *reduction in the number of both* H_{Ising} *pulses and bit flips as compared to* UNION-OF-STARS*, while maintaining Max-Cut approximation* ≥ 0.95.

Finally, we verify that our compilations reduce the amount of accumulated noise in two models. The first is a generic model for digital computations, where reducing the number of edges from m to $m' < m$ leads to an exponentially larger lower bound on the circuit fidelity, $F_0' = F_0^{m'/m}$. The second model considers the detailed physics of dephasing decoherence in trapped ions with global interactions, which is an important noise source in previous experiments with hundreds of trapped

ions [8]. We use a Lindbladian master equation to describe the effect of the dephasing noise, ignoring other noise sources, and derive an exact analytic expression for the expected cost in QAOA. The cost is exponentially suppressed with respect to the noiseless case, with an exponential prefactor that depends on the compilation time and a dephasing rate Γ. We analyze QAOA performance with varying Γ for our MQLib instances and compare the expected cost in the original compilation, with decomposition, and with decomposition and sparsification. We find decomposition has very significant benefits on maintaining a high-cost value in the presence of noise; sparsification is ineffective for this specific noise model, but we argue it would play an important role for noise sources that are not included in the present analytic treatment. We conclude our advanced compilation techniques are expected to provide significant benefits in reducing noise on quantum computing hardware, thus bridging some of the gap between classical computing and current quantum hardware for solving these problems.

We introduce some notation and give background on Max-Cut, QAOA, and UNION-OF-STARS in Section 2. We give our theoretical improvements for the number of H_{Ising} pulses and bit flips in Section 3 and corresponding experiments in 3.3. Section 4 discusses our noise model for QAOA and corresponding results. We conclude in Section 5. Omitted proofs and additional experiments are deferred to the appendices.

2 Preliminaries

We discuss the Max-Cut problem first and give background for the UNION-OF-STARS compilation. Next, we briefly review the background on classical sparsification algorithms.

Graphs and Max-Cut. For positive integer n, we denote $[n] = \{1, \ldots, n\}$ and $[0, n] = \{0, 1, \ldots, n\}$. We assume familiarity with basic definitions in graph theory; the interested reader is referred to [66] for a more detailed introduction. Graphs will be denoted by letters G, H, and are always simple (i.e., have no loops or multiple edges) and undirected. They may be weighted or unweighted. An unweighted graph $G = (V, E)$ is specified by the set of vertices V and edges E. We will assume that the vertex set $V = [n] := \{1, \ldots, n\}$ for an arbitrary but fixed positive integer n. The number of edges $|E|$ is denoted by m. All graphs are assumed to be connected so that $m \geq n - 1$. Further, $m \leq \frac{n(n-1)}{2} \leq \frac{n^2}{2}$ since all graphs are assumed to be simple. A weighted graph $G = (V, E, c)$ additionally has edge costs or weights $c : E \to \mathbb{R}_+$. An unweighted graph $G = (V, E)$ is equivalent to the weighted graph (V, E, c) with $c_e = 1$ for all $e \in E$. We assume that $c \geq 0$, i.e., all edge weights are nonnegative.

The *sum* of two graphs $G_1 = (V, E_1, c_1)$ and $G_2 = (V, E_2, c_2)$ is $G = (V, E_1 \cup E_2, c_1 + c_2)$. In particular, for an unweighted graph $G = (V, E)$ and a partition $(E_1, \ldots, E_T)$ of E, we have $G = \sum_{i \in [T]} G_i$ where $G_i = (V, E_i)$. For a constant $\alpha \in \mathbb{R}$, graph αG has edge costs multiplied with α, i.e., $\alpha G = (V, E, \alpha c)$. For example, in Fig. 2, we have $G = G_1 + 2G_2$.

Star graphs play a special role in the UNION-OF-STARS compilation of [50]. A star graph $G = (V, E)$ is an unweighted graph with a special vertex v^* called the central vertex and a subset $U \subseteq V \setminus \{v^*\}$ to which it is connected. That is, the only edges in G are of the form v^*u for $u \in U$. Any unweighted graph $G = (V, E)$ can be written as the sum of (at most) $n - 1$ star graphs; we omit the proof of this simple lemma:

Lemma 1. *Any unweighted graph $G = (V, E)$ can be expressed as a sum $\sum_{i \in [T]} G_i$ where each G_i is a star graph and $T \leq n - 1$.*

Given a graph $G = (V, E, c)$ and a vertex set $S \subseteq V$, the cut $\Delta_G(S)$ is the set of edges with exactly one endpoint in S, i.e., $\Delta_G(S) = \{uv \in E : |\{u, v\} \cap S| = 1\}$. The corresponding *cut value* is the sum of costs of edges in the cut, and is denoted as $\delta_G(S) = \sum_{e \in \Delta_G(S)} c_e$. For brevity, when G is clear from context, we denote this as $\delta(S)$. We often speak of 'cut S' when we mean 'cut $\Delta_G(S)$'. The Max-Cut in G is the cut corresponding to set $\arg\max_{S \subseteq V} \delta_G(S)$; the corresponding cut value of the maximum cut is $\delta_G(S)$ and we often denote it by $\delta_G^{\max}$.

Given G, computing $\delta_G^{\max}$ is NP-hard, i.e., there is no polynomial-time algorithm that given G computes $\delta_G^{\max}$ if P $\neq$ NP. For $\alpha \in [0,1]$, vertex set S is called an α-approximation for Max-Cut in G if $\delta_G(S) \geq \alpha \cdot \delta_G^{\max}$, where $\alpha = 1$ corresponds to the optimal cut. The celebrated Goemans-Williamson algorithm [21] is the classical state-of-the-art and gives a α^*-approximation for any graph G (in polynomial-time), where $\alpha^* \simeq 0.878$. Assuming the Unique Games Conjecture, no polynomial-time algorithm can achieve a better approximation ratio for all graphs [28]. Assuming P $\neq$ NP, no polynomial-time algorithm can achieve approximation ratio ≥ 0.942 for all graphs [24].

We refer to a cut with value at most half the total sum of all edge costs (i.e., $\frac{1}{2}\sum_{e\in E} c_e$) as *trivial*, and a cut with value $> \frac{1}{2}\sum_{e\in E} c_e$ as *non-trivial*. It is straightforward to obtain a cut with value $\frac{1}{2}\sum_{e\in E} c_e$, for example, using the greedy algorithm that iteratively adds or removes vertices from a cut S until its cut value can no longer be improved. Since $\sum_{e\in E} c_e$ is an upper bound on $\delta_G^{\max}$, this shows that obtaining a 0.5-approximation is straightforward.

Throughout, we reduce the Max-Cut problem on graph $G = (V, E, c)$ to a different graph $G' = (V, E', c')$ as follows: G' is constructed so that the cut value $\delta_G(S)$ of any non-trivial cut $S \subseteq V$ in G is $(1-\epsilon)$ within the cut value $\delta_{G'}(S)$ of S in G'. Therefore, solving Max-Cut in G' up to an α-approximation solves Max-Cut in G' up to an $\alpha(1-\epsilon)$-approximation in G, for any $\alpha \geq \frac{1}{2}$. In particular, obtaining the Max-Cut in G' gives a $(1-\epsilon)$-approximate cut in G.

Given graph $G = (V, E, c)$ with nonnegative edge weights, we use the QAOA variant that uses gates of the form $\exp(-i\gamma C)$ for the cost Hamiltonian $C = \sum_{uv\in E} c_{uv}\sigma_u^z\sigma_v^z$. For a vertex set $S \subseteq V$ and corresponding bit string $\boldsymbol{z}_S$, the expected values of $\langle C\rangle := \langle \boldsymbol{z}_S|C|\boldsymbol{z}_S\rangle$ and the cut value $\delta_G(S)$ are related as follows:

$$\begin{aligned}\langle C\rangle := \langle \boldsymbol{z}_S|C|\boldsymbol{z}_S\rangle &= \sum_{\substack{uv\in E:\\ |\{u,v\}\cap S|\neq 1}} c_{uv} - \sum_{\substack{uv\in E:\\ |\{u,v\}\cap S|=1}} c_{uv} \\ &= \sum_{uv\in E} c_{uv} - 2\delta_G(S). \end{aligned} \tag{3}$$

Therefore, the minimum eigenvector of C corresponds to the Max-Cut in G, which can be formulated as finding the maximum eigenvector of $\left(\frac{1}{2}\sum_{e\in E} c_e\right) I - \frac{1}{2}C$ [18].

Further, the lower $\langle \boldsymbol{z}_S|C|\boldsymbol{z}_S\rangle$ is, the larger the cut value $\delta_G(S)$. Since a non-trivial cut $S \subseteq V$ has cut value $\delta_G(S) > \frac{1}{2}\sum_{uv} c_{uv}$, we have that $\langle \boldsymbol{z}_S|C|\boldsymbol{z}_S\rangle < 0$ for non-trivial cuts S. This observation will be useful in Section 4, where we will compare a bit string $\boldsymbol{z}$ (generated using our algorithms) against baseline bit string $\boldsymbol{z}'$ using QAOA under noise in terms of their operator expectations $\langle \boldsymbol{z}|C|\boldsymbol{z}\rangle$, $\langle \boldsymbol{z}'|C|\boldsymbol{z}'\rangle$ to gauge their quality for Max-Cut.

Overview of union-of-stars. We next provide an overview of the UNION-OF-STARS algorithm [50] for generating cost Hamiltonians C for Max-Cut on trapped-ion devices using global H_{Ising} operations and bit flips. One approach to *compiling* these cost Hamiltonians $C = \sum_{uv\in E} c_{uv}\sigma_u^z\sigma_v^z$ on trapped-ion hardware is to start from the empty Hamiltonian $C = 0$ and apply a sequence of H_{Ising} pulses and bit flip operations. Each H_{Ising} pulse is implemented using a global MS operation and a pulse of length or time $t_p > 0$ adds all-to-all interaction terms $\sum_{u,v\in V} t_p\sigma_u^z\sigma_v^z$. The pulse may also be augmented with Pauli-X "bit flips" to change the sign of σ_v^z using $\sigma_v^x\sigma_v^z\sigma_v^x = -\sigma_v^z$. Using these "flips" on a set of vertices $S \subseteq V$ we obtain the general update rule for C:

$$C \leftarrow C \pm \sum_{u,v\in V} (-1)^{\mathbf{1}_S(u)+\mathbf{1}_S(v)} t_p\sigma_u^z\sigma_v^z, \tag{4}$$

where $\mathbf{1}_S(u)$ is the indicator for whether $u \in S$ and equals 1 if $u \in S$ and is 0 otherwise, while the sign $\pm$ of the update is a choice that can be made experimentally by changing the sign of the frequency of the laser driving the transition, as described in [50]. Formally, we define graph compilation as follows:

Definition 1 (Graph compilation)**.** *A sequence of H_{Ising} pulses and bit flip operations that produces graph coupling (as described above) for a graph G is called a graph compilation of G. $N(H_{\text{ISING}})(G)$ is*

the minimum number of H_{Ising} pulses and $N(\textsc{Total}_{\textsc{Ops}})(G)$ is the minimum number of total operations (H_{Ising} pulses and bit flips), in any graph compilation for G.

We say that the total time of the compilation for graph G is the sum of the lengths or times t_p of H_{Ising} pulses in the compilation[3]:

$$T = \sum_p t_p \tag{5}$$

Rajakumar *et al.* [50] conjecture that the problem of determining the shortest graph compilation for a given graph G is NP-hard, i.e., there may be no polynomial-time algorithm to find the compilation with the fewest number of H_{Ising} pulses. However, they give a polynomial-time algorithm to obtain a (possibly sub-optimal) compilation with the number of H_{Ising} pulses upper bounded linearly in the input size: at most $3n-2$ for unweighted graphs and at most $3m+1$ for weighted graphs.

They start by noting that the graph $G_1 + G_2$ can be compiled by combining the compilations for each G_i $(i = 1, 2)$ first:

Lemma 2 ([50], Lemma 1)**.** *Suppose $G = \alpha_1 G_1 + \alpha_2 G_2$ for weighted graphs G_1, G_2 on vertex set V and reals $\alpha_1, \alpha_2 \in \mathbb{R}$. Then, if there exist graph compilations with s_i H_{Ising} pulses and t_i bit flips for graph G_i, $i \in \{1, 2\}$, then there exists a graph compilation for G with $s_1 + s_2$ H_{Ising} pulses and $t_1 + t_2$ bit flips. In particular, $N(H_{\textsc{Ising}})(G) \leq N(H_{\textsc{Ising}})(G_1) + N(H_{\textsc{Ising}})(G_2)$ and $N(\textsc{Total}_{\textsc{Ops}})(G) \leq N(\textsc{Total}_{\textsc{Ops}})(G_1) + N(\textsc{Total}_{\textsc{Ops}})(G_2)$.*

They also show the existence of compilations for all graphs and give a polynomial-time algorithm called UNION-OF-STARS that gives a specific sequence of H_{Ising} pulses and bit flips for a given graph. Their construction observes that an unweighted star graph can be compiled using 4 pulses. By Lemma 1, this implies that an unweighted graph on n vertices can be compiled using at most $4(n-1)$ pulses, or $N(H_{\text{Ising}})(G) \leq 4(n-1)$. Further, since any single edge is also a star graph, they decompose any weighted graph with m edges into its edges, and obtain $N(\textsc{Total}_{\textsc{Ops}})(G) \leq 4m$. By combining redundant pulses, they improve these numbers to $3(n-1)+1$ and $3m+1$ respectively:

Theorem 1 ([50], Theorems 5, 6)**.** *$N(H_{\textsc{Ising}})(G) \leq 3n-2$ for an unweighted graph and $N(H_{\textsc{Ising}})(G) \leq 3m+1$ for a weighted graph.*

The following result for $N(\textsc{Total}_{\textsc{Ops}})$ is implicit in their construction:

Lemma 3. *For any graph G with m edges, $N(\textsc{Total}_{\textsc{Ops}})(G) = O(m)$.*

Algorithm 1: GRAPH-SPARSIFICATION-USING-EFFECTIVE-RESISTANCES [58]

Input: weighted graph $G = (V, E, c)$ and parameter $\epsilon > 0$
Output: sparsifier $H = (V, E', c')$ of G

compute effective resistances r_e for each $e \in E$
set $q = \frac{C_0 n \log n}{\epsilon^2}$ for a large enough constant C_0
for $t = 1$ to q, independently **do**:
    choose a random edge $e \in E$ with probability p_e proportional to $c_e r_e$
    add e to H with weight $\frac{c_e}{q p_e}$, summing weights if e is already in H
return H

[3]We omit the time taken by bit flip operations since they are at least an order of magnitude smaller in comparison to H_{Ising} pulses [50].

Graph sparsification. The theory of graph sparsifiers was initiated by Karger in 1994 [25]. Given any input graph $G = (V, E, c)$ on n vertices and an error parameter $\epsilon > 0$, graph $H = (V, E', c')$ is called a *graph* sparsifier of G if (a) G, H have similar cut values, i.e., for each $S \subseteq V$, the cut value $\delta_G(S)$ is within factor $1 \pm \epsilon$ of the cut value $\delta_H(S)$ and (b) H has at most $O\left(n \log^{\kappa} n/\epsilon^2\right)$ edges for some constant κ. Note that $\delta_H(S)$ is computed with respect to the edge cost function c' of H.

The current best provable graph sparsification algorithms can reduce the number of edges to almost linear in the vertices, with some loss in approximation for the cut values of the graph, *for all cuts in the graph*:

Theorem 2 ([4], Theorem 1.1)**.** *There exists a polynomial-time algorithm that given a weighted graph $G = (V, E, c)$ on n vertices and error parameter $\epsilon > 0$, finds another weighted graph $H = (V, E', c')$, such that with high probability*[1]

(i) All cut values are preserved, i.e., $1 - \epsilon \leq \frac{\delta_H(S)}{\delta_G(S)} \leq 1 + \epsilon, \quad \forall S \subseteq V,$

(ii) H is sparse, i.e., $|E'| = O\left(\frac{n}{\epsilon^2}\right)$.

In other words, if the Max-Cut was found on the sparse graph H, which can be computed classically, then it would approximate the Max-Cut on G with $(1 - \epsilon)$-approximation. Note that this result approximates *all* the cuts in the graph[4]. The algorithm of [4] is significantly involved, and therefore, for the (classical) experiments in this work, we will instead use the simpler and more efficient sparsification algorithm of [58] based on computing effective resistances[2]. This algorithm outputs sparsifiers with a slightly weaker guarantee of $|E'| = O\left(\frac{n \log n}{\epsilon^2}\right)$ for the number of edges. We state it in Algorithm GRAPH-SPARSIFICATION-USING-EFFECTIVE-RESISTANCES [58] for completeness (but without proof). We will use these sparsification results as black boxes and modify the UNION-OF-STARS compilation to provide guarantees on $N(H_{\text{ISING}})$ and $N(\text{TOTAL}_{\text{OPS}})$ accordingly.

3 Trapped-ion compilations using sparse UNION-OF-STARS

In Section 3.1, we present two variants of our decomposition algorithm and reduce the number of H_{Ising} pulses required in UNION-OF-STARS (i.e. $N(H_{\text{ISING}})$) for weighted graphs from $O(m)$ to $O(n \log(n/\epsilon))$. Both decomposition algorithms use the facts that (1) removing edges with very small edge weight does not change the cut value of large cuts, and (2) the remaining graph can be written as the weighted sum of a small number of unweighted graphs. In Section 3.2, we improve the total number of gates (i.e., H_{Ising} pulses and bit flips or $N(\text{TOTAL}_{\text{OPS}})$) for all graphs from $O(m)$ to $O\left(\frac{n \log(n/\epsilon)}{\epsilon^2}\right)$. We do this by first decomposing the graph to reduce $N(H_{\text{ISING}})$, and the applying sparsification to reduce the number of edges in the decomposed graph. In Section 3.3, we run experiments on the MQLib graph library and show that our algorithms significantly outperform UNION-OF-STARS in classical simulations (assuming no noise).

3.1 Reduction in number of H_{Ising} pulses for weighted graphs

First, we note that improving $N(H_{\text{ISING}})(G)$ from $O(m)$ to $O\left(\frac{n}{\epsilon^2}\right)$ is an immediate consequence of sparsification (Theorem 2) and the UNION-OF-STARS upper bound (Theorem 1). In this section, we give two algorithms to improve the dependence of $N(H_{\text{ISING}})(G)$ on ϵ,

1. Algorithm EXP-DECOMPOSE , which improves it to $O(\frac{n}{\epsilon} \log\left(\frac{n}{\epsilon}\right))$ and
2. Algorithm BINARY-DECOMPOSE , which further improves it to $O\left(n \log\left(\frac{n}{\epsilon}\right)\right)$.

[4] There is another stream of literature that focuses on sparsifying the set of vertices in the graph, so that the minimum cuts across a given smaller set of terminals is preserved [38, 10]. However, it is unclear how this can be used to obtain guarantees for approximating the Max-Cut.

Algorithm 2: EXP-DECOMPOSE

Input: weighted graph $G = (V, E, c)$ and parameter $\epsilon > 0$

Output: weighted graph G' such that $1 - \epsilon \leq \frac{\delta_{G'}^{\max}}{\delta_G^{\max}} \leq 1$

set $c^* = \max_{e \in E} c(e)$
set $\tau = \frac{\epsilon c^*}{2n^2}$ and set $k = \lceil \log_{1+\epsilon/2}(c^*/\tau) \rceil$
for $j \in [0, k]$ **do**
    initialize undirected graph $G_j = (V, E_j)$ with $E_j = \emptyset$
for each edge $e \in E$ **do**
    if $c(e) > \tau$ **then**
        let j be the unique integer in $[k]$ such that $\tau(1+\epsilon/2)^{j-1} < c(e) \leq \tau(1+\epsilon/2)^j$
        add edge e to E_j
return weighted graph $G' = (V, E', c')$ defined as

$$G' = \sum_{j \in [k]} (1 + \epsilon/2)^{j-1} G_j.$$

This latter bound is logarithmic in $1/\epsilon$ and is therefore an *exponential* improvement over $O\left(\frac{n}{\epsilon^2}\right)$.

Both algorithms work as follows: first, we decompose the given weighted graph $G = (V, E, c)$ into unweighted graphs $G_1, \ldots, G_k$ and weights $\alpha_1, \ldots, \alpha_k > 0$ such that $G \simeq \sum_{j \in [k]} \alpha_j G_j$ (we shortly define $\simeq$). Since each G_j is unweighted, it takes at most $O(n)$ H_{Ising} pulses to compile using UNION-OF-STARS (see Theorem 1). Then, we combine these compilations using Lemma 3 into a single compilation for G that takes $O(kn)$ H_{Ising} pulses. For Algorithm EXP-DECOMPOSE , $k = O\left(\frac{1}{\epsilon} \log\left(\frac{n}{\epsilon}\right)\right)$ while Algorithm BINARY-DECOMPOSE improves this to $k = O\left(\log \frac{n}{\epsilon}\right)$. We begin with the description and formal guarantee for Algorithm EXP-DECOMPOSE , followed by Algorithm BINARY-DECOMPOSE . All proofs are deferred to Section A.

Algorithm exp-decompose . Intuitively, the algorithm obtains graphs $G_1, \ldots, G_k$ by grouping edges in G that have similar weights (within a factor $\simeq (1 + \epsilon/2)$). Crucially, edges with very small costs are removed entirely, as follows: denote the max cost as $c^* = \max_{e \in E} c(e)$, then all edges with costs smaller than the *threshold* $\tau := \frac{\epsilon c^*}{2n^2}$ are removed. Since the total cost of such edges is at most $m \times \tau \leq \frac{n^2}{2} \times \frac{\epsilon c^*}{2n^2} = \frac{\epsilon}{4} c^* \leq \frac{\epsilon}{4} \delta_G^{\max}$, this reduces the Max-Cut in G by factor at most $(1 - \epsilon/4)$. The details are presented in Algorithm EXP-DECOMPOSE . We present the theoretical guarantee of the algorithm next:

Theorem 3. *Given a weighted graph $G = (V, E, c)$ and error parameter $\epsilon \in (0, 1)$, EXP-DECOMPOSE returns another weighted graph $G' = (V, E', c')$ such that we get*

(i) Preservation of non-trivial cuts, i.e., for all $S \subseteq V$ with $\delta_S(G) \geq \frac{1}{2} \sum_{e \in E} c_e$,

$$1 - \epsilon \leq \frac{\delta_{G'}(S)}{\delta_G(S)} \leq 1.$$

In particular, the Max-Cut in G' is a $(1 - \epsilon)$-approximate cut in G.

(ii) Reduction in H_{Ising} pulses: $N(H_{ISING})(G') = O\left(\frac{n}{\epsilon} \log \frac{n}{\epsilon}\right)$.

Further, the number of edges in G' is at most the number of edges in G.

Algorithm 3: BINARY-DECOMPOSE

Input: weighted graph $G = (V, E, c)$ and parameter $\epsilon > 0$

Output: weighted graph G' such that $1 - \epsilon \leq \frac{\delta_{G'}^{\max}}{\delta_G^{\max}} \leq 1$

set $c^* = \max_{e \in E} c(e)$ and $\eta = \frac{\epsilon c^*}{n^2}$

set $k = 1 + \lfloor \log_2(n^2/\epsilon) \rfloor$

for each edge $e \in E$ **do**

    set $d(e) = \lfloor \frac{c(e)}{\eta} \rfloor$

    write $d(e) \in [0, n^2/\epsilon]$ in binary with k digits

$$b_{k-1}(e) b_{k-2}(e) \dots b_0(e)$$

for $j \in [k]$ **do**

    let $G_j = (V, E_j)$ be the graph such that $e \in E_j$ iff $b_{j-1}(e) = 1$

return weighted graph $G' = (V, E', c')$ defined as

$$G' = \eta 2^{k-1} G_k + \eta 2^{k-2} G_{k-1} + \ldots + \eta 2^0 G_1.$$

Algorithm binary-decompose . Our second algorithm decomposes $G \simeq \sum_{j \in [k]} \alpha_j G_j$ by first computing the binary representation of each edge cost $c(e)$ up to k digits. The jth unweighted graph G_j contains exactly those edges with the jth bit in their binary representation equal to 1, and the weights α_j increase in powers of 2. For error parameter $\epsilon > 0$, we choose $k = 1 + \lfloor \log_2(n^2/\epsilon) \rfloor$ to ensure that Max-Cut in decomposition $\sum_{j \in [k]} \alpha_j G_j$ is within factor $(1 + \epsilon)$ of the Max-Cut in G (see proof details in Appendix A). The details are presented in Algorithm BINARY-DECOMPOSE and the next theorem contains the formal guarantee:

Theorem 4. *Given a weighted graph $G = (V, E, c)$ and error parameter $\epsilon \in (0, 1)$, BINARY-DECOMPOSE returns another weighted graph $G' = (V, E', c')$ such that we get*

(i) Preservation of non-trivial cuts, i.e., for all $S \subseteq V$ with $\delta_S(G) \geq \frac{1}{2} \sum_{e \in E} c_e$,

$$1 - \epsilon \leq \frac{\delta_{G'}(S)}{\delta_G(S)} \leq 1.$$

In particular, the Max-Cut in G' is a $(1 - \epsilon)$-approximate cut in G.

(ii) Reduction in H_{Ising} pulses: $N(H_{\text{ISING}})(G') = O\left(n \log \frac{n}{\epsilon}\right)$.

Further, the number of edges in G' is at most the number of edges in G.

These algorithms have a significant impact on the reduction of the number of H_{Ising} pulses in the compilation. This will be evident from our classical experiments that simply count these operations for the modified compilation.

3.2 Reduction in total number of operations

BINARY-DECOMPOSE and EXP-DECOMPOSE reduce $N(H_{\text{ISING}})$ for weighted graphs, but they may not reduce the total number of operations which also involve bit flips. Recall that $N(\text{TOTAL}_{\text{OPS}})(G)$ is upper bounded by $O(m)$ for all graphs with m edges. We combine our decomposition idea with sparsification to reduce the number of edges (and hence the $N(\text{TOTAL}_{\text{OPS}})$) while still maintaining $N(H_{\text{ISING}}) = O(n \log(n/\epsilon))$.

Algorithm 4: SPARSE-UNION-OF-STARS

Input: weighted graph $G = (V, E, c)$ and parameters $\epsilon_1, \epsilon_2 > 0$
Output: weighted graph $G = (V, E', c')$
1: $H =$ GRAPH-SPARSIFICATION-USING-EFFECTIVE-RESISTANCES (G, ϵ_1)
2: $G'_{\text{bin}} =$ BINARY-DECOMPOSE(H, ϵ_2) and $G'_{\text{exp}} =$ EXP-DECOMPOSE(H, ϵ_2)
3: set

$$G' := \begin{cases} G'_{\text{bin}} & \text{if } N(H_{\text{ISING}})(G'_{\text{bin}}) < N(H_{\text{ISING}})(G'_{\text{exp}}) \\ G'_{\text{exp}} & \text{if } N(H_{\text{ISING}})(G'_{\text{bin}}) \geq N(H_{\text{ISING}})(G'_{\text{exp}}) \end{cases}$$

4: **return** G' and UNION-OF-STARS(G')

Theorem 5. *Given a weighted graph $G = (V, E, c)$ and error parameter $\epsilon \in (0, 1)$, SPARSE-UNION-OF-STARS with $\epsilon_1 = \epsilon_2 = \frac{\epsilon}{3}$ returns another weighted graph $G' = (V, E', c')$ such that with high probability, we get*

(i) Preservation of non-trivial cuts, i.e., for all $S \subseteq V$ with $\delta_S(G) \geq \frac{1}{2}\sum_{e \in E} c_e$,

$$1 - \epsilon \leq \frac{\delta_{G'}(S)}{\delta_G(S)} \leq 1 + \epsilon.$$

In particular, the Max-Cut in G' is a $(1-\epsilon)$-approximate cut in G.

(ii) Reduction in H_{Ising} Pulses: $N(H_{\text{ISING}})(G') = O\left(n \log \frac{n}{\epsilon}\right)$,

(iii) Reduction in total number of operations: $N(\text{TOTAL}_{\text{OPS}})(G') = O\left(\frac{n \log(n/\epsilon)}{\epsilon^2}\right)$.

Note that achieving a graph G' with guarantee (i) and with $N(\text{TOTAL}_{\text{OPS}}) = O(n/\epsilon^2)$ follows directly from Lemma 3 and Theorem 2. However, having an additional guarantee on $N(H_{\text{ISING}})$ requires the decomposition technique. The detailed proof is presented in the proof in Appendix A.

Choice of the decomposition algorithm. Note that in Algorithm SPARSE-UNION-OF-STARS , we choose *the best of* EXP-DECOMPOSE and BINARY-DECOMPOSE , i.e., whichever requires a smaller number of H_{Ising} pulses. Since both these algorithms are classical, this is not computationally prohibitive for QAOA. This is useful since EXP-DECOMPOSE often performs better than its *worst-case* theoretical guarantee from Theorem 3 suggests, particularly for smaller graphs and larger values of error parameter ϵ_2. We emphasize that this is not always the case and Figure 7 shows an example of a large graph ($\simeq 400,000$ edges) where BINARY-DECOMPOSE performs better. Appendix B discusses this in greater detail.

3.3 Compilations using SPARSE-UNION-OF-STARS on the MQLib graph library

To exemplify the impact of sparsification and decomposition on the number of operations in graph compilation, we classically simulate the Algorithm SPARSE-UNION-OF-STARS on graphs from the MQLib library [14], a collection of challenging graphs for the Max-Cut problem designed to benchmark Max-Cut algorithms[5]. To keep the computation of exact Max-Cut manageable, we choose all positive weighted graphs in MQLib with $50 \leq n < 200$ and $2n \leq m < 2000$. The condition $m \geq 2n$ excludes very sparse graphs (where edge-by-edge compilations are inexpensive anyway), resulting in 161 graphs in total. We refer to the original graph as G and the corresponding modified instance as G'. We compare vanilla UNION-OF-STARS and SPARSE-UNION-OF-STARS for various combinations of sparsification and decomposition parameters on three metrics: (1) number of H_{Ising} pulses or $N(H_{\text{ISING}})(H)$ value, (2) number of total gates or $N(\text{TOTAL}_{\text{OPS}})(H)$ value, and (3) total compilation time T (see eqn. (5)), where $H \in \{G, G'\}$.

[5]The source code for computations in this section can be found at https://github.com/jaimoondra/union-of-stars [40].

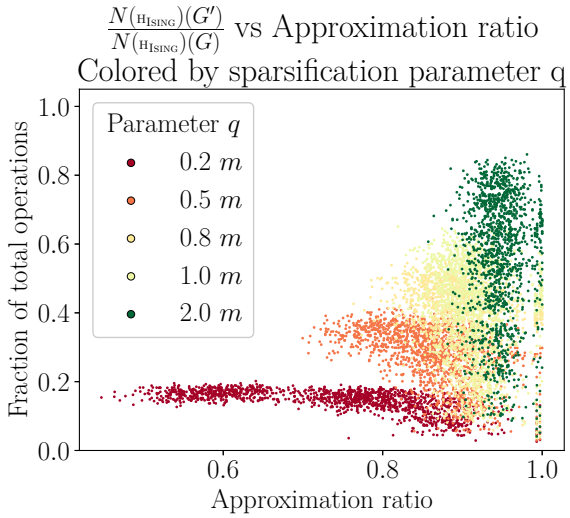


Figure 3: Reduction in $N(H_{\text{ISING}})$ (total number of H_{Ising} pulses) vs Max-Cut approximation for various runs of our experiment on weighted graphs in MQLib, colored by parameter q. Each data point is a single run. Points on the lower right have high reductions and high Max-Cut approximation.

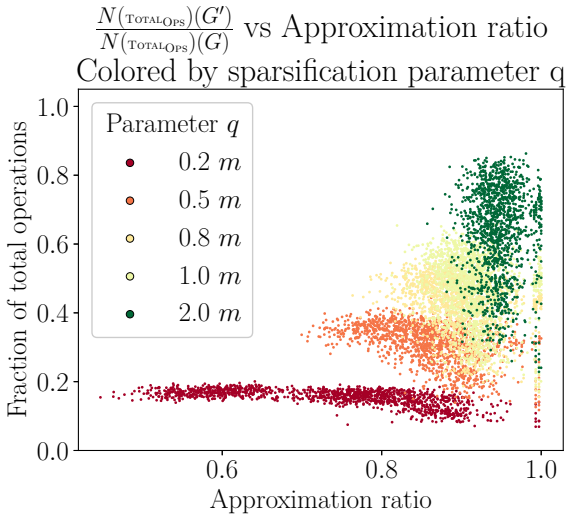


Figure 4: Reduction in $N(\text{TOTAL}_{\text{OPS}})$ (total number of H_{Ising} pulses and bit flip gates) vs Max-Cut approximation for various runs of our experiment on weighted graphs in MQLib. Each data point is a single run. Points on the lower right have high reductions and high Max-Cut approximation.

Table 2: Average reduction in H_{Ising} pulses, total operations, and compilation time for parameter combinations of sparsification and decomposition that yield an average Max-Cut approximation ratio $\frac{\delta^{\max}_{G'}}{\delta^{\max}_{G}} \geq 0.90$ across graphs in the MQLib library (with $50 \leq n < 200$ and $2n \leq m < 2000$), using Algorithm SPARSE-UNION-OF-STARS . Here G is the original input graph, and G' is the decomposed + sparsified instance. Smaller numbers are better for $N(H_{\text{ISING}}), N(\text{TOTAL}_{\text{OPS}})$, and compilation time columns. Numbers closer to 1 are better for the ratio of the Max-Cut approximation column. Smaller q gives more sparsified instances.

Sparsification parameter q	Decomposition parameter ϵ_2	$\frac{N(H_{\text{ISING}})(G')}{N(H_{\text{ISING}})(G)}$	$\frac{N(\text{TOTAL}_{\text{OPS}})(G')}{N(\text{TOTAL}_{\text{OPS}})(G)}$	$\frac{\text{compilation time of } G'}{\text{compilation time of } G}$	Max-Cut approximation for G attained using the Max-Cut in G'
0.8	1.00	0.352	0.569	0.404	0.901
1.0	0.10	0.535	0.870	0.557	0.915
1.0	0.25	0.488	0.780	0.523	0.917
1.0	0.50	0.443	0.681	0.491	0.914
1.0	0.75	0.419	0.614	0.474	0.916
1.0	1.00	0.400	0.567	0.460	0.916
1.0	2.00	0.351	0.439	0.425	0.913
1.0	5.00	0.301	0.299	0.389	0.912
2.0	0.10	0.733	0.873	0.762	0.949
2.0	0.25	0.668	0.774	0.716	0.949
2.0	0.50	0.605	0.668	0.670	0.948
2.0	0.75	0.561	0.594	0.638	0.946
2.0	1.00	0.530	0.537	0.616	0.945
2.0	2.00	0.454	0.406	0.562	0.947
2.0	5.00	0.364	0.266	0.498	0.944

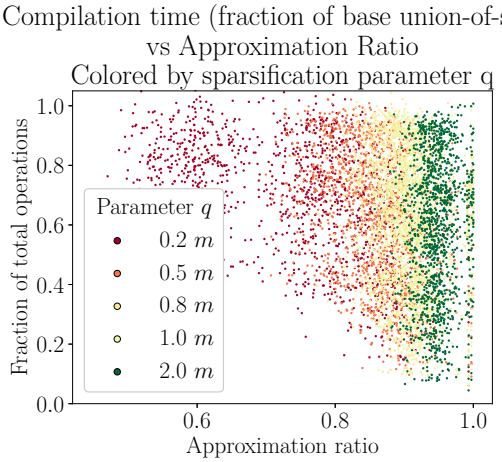


Figure 5: Reduction in the time $T = \sum_p t_p$ of compilation vs Max-Cut approximation for various runs of our experiment on weighted graphs in MQLib. Each data point is a single run.

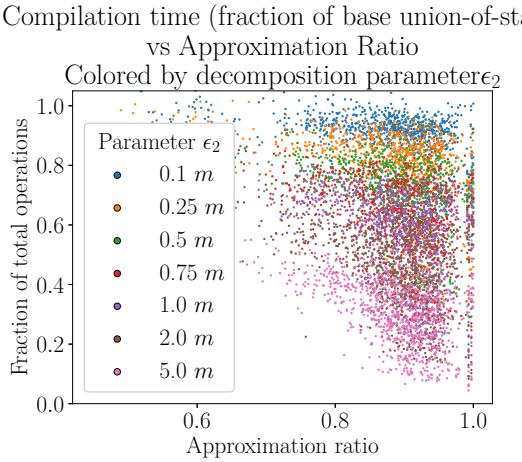


Figure 6: Reduction in the time $T = \sum_p t_p$ of compilation vs Max-Cut approximation for various runs of our experiment on weighted graphs in MQLib. Each data point is a single run.

Choice of algorithm parameters. We parameterize each run by q that denotes the number of edges sampled from the original graph G to get sparsified instance G' in Algorithm GRAPH-SPARSIFICATION-USING-EFFECTIVE-RESISTANCES [58] ; recall that q and error parameter ϵ_1 are related as $q = \frac{Cn \log n}{\epsilon_1^2}$ for a suitable constant C. For each graph G with m edges, we run SPARSE-UNION-OF-STARS for various combinations of $q \in \{0.2m, 0.5m, 0.8m, 1.0m, 2.0m\}$ and $\epsilon_2 \in \{0.1, 0.25, 0.5, 0.75, 1.0, 2.0, 5.0\}$. Lower values of q correspond to greater sparsification and higher values of ϵ_2 correspond to stronger effect of decomposition. Note that while our theoretical guarantees for decomposition algorithms (Theorem 3, 4) assume that $\epsilon_2 \in (0, 1)$, in practice higher values of ϵ_2 perform very well while retaining a high approximation ratio.

Results. Fig. 3 shows the fractional reduction $\frac{N(H_{\text{ISING}})(G')}{N(H_{\text{ISING}})(G)}$ using SPARSE-UNION-OF-STARS as compared to UNION-OF-STARS, plotted against the Max-Cut approximation of G' for G. Each data point represents one run of the experiment on a specific graph and with specific values of parameters q and ϵ_2. For most graphs G, we observe over 40% reduction in $N(H_{\text{ISING}})$ value, while still recovering over 90% of the Max-Cut in the original graph for suitable choices of the parameters (e.g., points in purple and red). Further, there is a clear trade-off in the Max-Cut approximation and the fractional reduction $\frac{N(H_{\text{ISING}})(G')}{N(H_{\text{ISING}})(G)}$; higher reduction necessarily reduces the quality of the approximation. There is also a clear dependence on q: sparser graphs lead to a greater reduction in $N(H_{\text{ISING}})$ value but obtain poorer approximation. Similar results hold for the total number of operations (Fig. 4).

Similar reductions also hold for the total time of pulses. This is shown in Figs. 5 and 6; the figures are identical except they are colored by the sparsification parameter q and the decomposition parameter ϵ_2, respectively. In Fig. 6 observe that decomposition has a significant impact in the reduction of the total time of pulses, while sparsification (Fig. 5) does not have any significant correlation. We conclude from Figs. 3-6 that sparsification reduces the total number of operations, while decomposition reduces the total time of pulses.

The total compilation time T does not depend on the sparsification parameter q because, in edge-by-edge compilations, T is determined by the total weight of the edges in the graph–which remains largely unaltered by sparsification. As an example, consider a complete bipartite graph G on vertex set $A \cup B$ with $|A| = |B| = n/2$ with edge set $E(G) = \{uv : u \in A, v \in B\}$ with each edge weight $= 1$. Then it can be shown that the following (weighted) graph H on vertex set $A \cup B$ is a sparsifier of G: for each

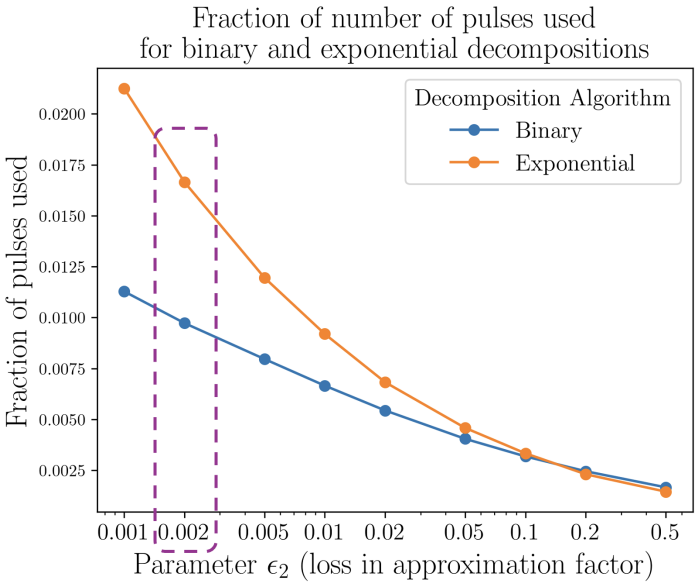


Figure 7: At a loss of ϵ_2 in the approximation factor, the fraction of H_{Ising} pulses used by BINARY-DECOMPOSE and EXP-DECOMPOSE for the weighted graph g001737 from MQLib with $n = 952$ vertices and $m = 430,730$ edges is depicted in blue and orange respectively. Compared to the edge-by-edge UNION-OF-STARS construction, BINARY-DECOMPOSE and EXP-DECOMPOSE use 1% and 1.75% pulses respectively, while maintaining a 0.998-approximation factor for Max-Cut for the original graph (corresponding to $\epsilon_2 = 0.002$, highlighted in purple). The gap between the performance of the algorithms reduces as the loss ϵ_2 of the approximation factor is increased. This pattern holds for other large and dense graphs in MQLib, see Appendix B for more examples.

$u \in A, v \in B$, include edge $uv \in E(H)$ with probability $p = \frac{1}{\sqrt{n}}$ independent of other edges. Assign weight $\frac{1}{p} = \sqrt{n}$ to this edge. In an edge-by-edge construction, the total length of pulses in G is the sum of edge weights (up to constants), which is $\simeq n^2/4$, the same as for H. However, the number of pulses for G is $O(|E(G)|) = O(n^2)$ while the number of pulses for H is $O(|E(H)|) = O(p|E(G)|) = O(n^{3/2})$.

4 Quantum noise analysis

Here, we consider the expected decrease in noise during quantum computations that utilize graph sparsification and decomposition techniques for compiling MaxCut with QAOA. We first present a device-agnostic theoretical analysis for compiling sparsified instances with digital quantum computers using local one- and two-qubit gates, then analyze the expected noise in analog quantum computations with decomposition and sparsification.

4.1 Sparsified compilations of QAOA for digital quantum computations

Digital quantum computations are compiled to hardware-native universal gate sets that act on one or two qubits at a time. This applies to superconducting qubits [9], certain rydberg atom devices [7], trapped ions implementing two-qubit gates [47], and any other digital quantum computing technology. For concreteness we consider a hardware gate set containing controlled-not $\text{CNOT}_{uv} = U_{uv}$ operators as well as generic single-qubit rotations $\exp(-i\theta\sigma_v^\alpha) = U_v^\alpha(\theta)$ where $\alpha \in \{x, y, z\}$, though similar considerations apply to other hardware gates sets. We do not consider specific hardware connectivities or SWAP gate requirements, but instead assume that each qubit may interact with each other qubit, to keep the discussion generic.

The QAOA unitary operator $\exp(-i\gamma C) = \prod_{(u,v)\in E} \exp(-i\gamma\sigma_u^z\sigma_v^z)$ is compiled from component operations $\exp(-i\gamma\sigma_u^z\sigma_v^z)$ for each edge in the graph. Each edge operation is compiled into our native gate set as

$$\exp(-i\gamma\sigma_u^z\sigma_v^z) = U_{uv}U_v^z(\gamma)U_{uv}. \tag{6}$$

For a quantum state described by a density operator ρ, the ideal evolution under any operator U is

$$\rho' = U\rho U^\dagger \tag{7}$$

where $\dagger$ denotes the Hermitian conjugate. Applying component gates in sequence we obtain the desired unitary dynamics under the coupling operator unitary $\rho \leftarrow e^{-i\gamma C}\rho e^{i\gamma C}$.

Noisy quantum circuit operations can be described in the quantum channel formalism using Krauss operators $\sqrt{p_k}E_k$ with $\sum_k p_k E_k^\dagger E_k = \mathbb{1}$, with $\mathbb{1}$ the identity operator, $p_k \geq 0$, and $\sum_k p_k = 1$ [43]. Analogous to (7) this gives

$$\rho' = \sum_k p_k E_k U \rho U^\dagger E_k^\dagger, \tag{8}$$

where we have used a notation in which the intended dynamics under U is separated from the E_k. Equation (8) is mathematically equivalent to a probabilistic model where evolution $E_k U$ is generated with probability p_k. We consider $E_1 = \mathbb{1}$ as the ideal evolution, with probability p_1, while $E_k \neq \mathbb{1}$ for $k \geq 2$ represent various types of noisy evolution with probabilities p_k. Applying a sequence of gates $U^{(i)}$ to compile the graph coupling operator $\exp(-i\gamma C) = \prod_{i=1}^{N_{\text{gates}}} U^{(i)}$, the final state after these circuit operations can be expressed as

$$\rho_{\text{final}} = F_0\rho_{\text{ideal}} + (1 - F_0)\rho_{\text{noise}} \tag{9}$$

where

$$F_0 = \prod_{i=1}^{N_{\text{gates}}} p_1^{(i)} \tag{10}$$

is the probability of generating ideal evolution throughout the entire circuit. It can be shown that F_0 is a lower bound to the quantum state fidelity, from which we can derive an upper bound $M = \log(1-P)/\log(1-F_0)$ on how many measurements M are necessary to sample a result from the ideal quantum probability distribution with probability P [33]. For example, if the ideal circuit prepared the optimal solution, then in the absence of noise a single measurement would provide this optimal solution. In the presence of noise, if we wanted to sample this solution with probability P, then we would need at most M measurements.

We are now ready to consider how sparsification is expected to influence noise in digital quantum circuits. As a first approximation, we can consider that all two-qubit gates U_{uv} have identical probabilities for ideal evolution $\overline{p}_1$, and we can neglect single-qubit gate errors, since these are typically much smaller than two-qubit gate errors. A more precise treatment can take $\overline{p}_1$ as the geometric mean of two-qubit gate errors. In either case we have $F_0 = \overline{p}_1^{2m}$ for a graph with m edges, since each edge requires two two-qubit gates in our compilation. For dense instances m may scale quadratically with n while for sparsified instances with a constant error tolerance ϵ the number of sparsified edges $m' = O(n\log n)$ scales nearly linearly with n. The fidelity lower bounds for these circuits satisfy $F_0' = F_0^{m'/m}$, which may be a very significant improvement in cases where $m' \ll m$. Minimizing the number of noisy operations through sparsification is therefore expected to increase the circuit fidelity and to reduce the number of measurements needed for a high quality result.

4.2 Sparsified and decomposed compilations of QAOA for analog quantum computation

Here we consider sparsification and decomposition in compilations for analog trapped-ion quantum hardware, with a specific noise model, as follows. We consider an n-ion quantum state evolving continuously in time under a native optical-dipole-force interaction described by an effective Ising

Hamiltonian $H = n^{-1}H_{\text{Ising}}$ [8]; the scaling $\sim n^{-1}$ is a realistic feature that arises from coupling to the center-of-mass vibrational mode. The σ_u^x operations are implemented through π pulses, which we approximate as instantaneous and noiseless. For the evolution under H we model noise as dephasing on each qubit at rate Γ. Such dephasing is present in analog trapped ion experiments, due to fundamental Rayleigh scattering of photons as well as technical noise sources, which contributes significantly to single-qubit decoherence in experiments such as Refs. [8, 62]. This is one of several sources of noise that are present in large-scale trapped ion experiments, and we choose this particular noise model because it is amenable to an exact analytic treatment of single-layer QAOA compiled with UNION-OF-STARS, with or without sparsification and decomposition, as described further in Appendix C.

We emphasize that our dephasing noise model does not capture all sources of noise that are relevant in trapped ion experiments. Additional sources of noise, such as Raman light scattering transitions, laser power fluctuations, electron-vibration coupling, and state preparation and measurement errors are also important in first-principles physical models of trapped ion dynamics [20, 31, 34, 39]. For example, Mølmer-Sørensen interactions are often timed or controlled so that the periodic vibrational modes return to their initial states and become disentangled from the electrons after the interaction [6, 63]. However the complexity of the time-dependent vibrational spectrum presents challenges for gate timing and control, which can lead to undesired entanglement between the electronic and vibrational degrees of freedom with a decreased purity of the computational state [31]. This source of noise could be expected to decrease computational fidelity in a way that depends on the number of applications of H in the graph compilation, since each application of H provides an opportunity for generating vibration-electron entanglement. Methods to address these and other error sources are topics of ongoing research [39, 63]. However, treating these additional sources of noise analytically in the present context requires a considerably more complicated analysis that is beyond our scope here. We instead focus on dephasing noise only, which we are able to treat analytically in single-layer QAOA. Although this is only a simplified treatment, it nonetheless enables us to understand some experimentally relevant effects of noise for large instances and deep compilations, which would not be possible with detailed physics simulations of all relevant noise sources.

We model the continuous-time quantum evolution using the Lindbladian master equation

$$\frac{d\rho}{dt} = -i(H\rho - \rho H) - \sum_u (J_u J_u^\dagger \rho - 2J_u \rho J_u^\dagger + \rho J_u J_u^\dagger). \tag{11}$$

Here $-i(H\rho - \rho H)$ represents the ideal (Schrödinger) quantum state evolution while the $J_u = \sqrt{\Gamma/8}\sigma_u^z$ represent noise due to dephasing, and $H = n^{-1}H_{\text{Ising}} = n^{-1}\sum_{i<j}\sigma_i^z\sigma_j^z$ is the effective Hamiltonian for the optical-dipole-force detuned close to the n-ion center-of-mass mode [8, 20]. Using the techniques from Refs. [20, 34], we derived the cost expectation value for single-layer QAOA with problem graph coupling operator $C = \sum_{(u,v)\in E} c_{uv}\sigma_u^z\sigma_v^z$ and with a potentially different graph coupling operator $C' = \sum_{(u,v)\in E'} c'_{uv}\sigma_u^z\sigma_v^z$ in compilation, which can represent the approximate coupling operators used in sparsification or decomposition. The cost expectation value is

$$\begin{aligned}\langle C\rangle = &\sum_{u<v} \frac{c_{uv}\sin(4\beta)\sin(2\gamma(t)c'_{uv})e^{-\Gamma t/2}}{2}\left(\prod_{\mu\neq u,v}\cos(2\gamma(t)c'_{\mu v}) + \prod_{\mu\neq u,v}\cos(2\gamma(t)c'_{\mu u})\right)\\ &- \sum_{u<v}\frac{c_{uv}\sin^2(2\beta)e^{-\Gamma t}}{2}\left(\prod_{\mu\neq u,v}\cos(2\gamma(t)(c'_{\mu u}+c'_{\mu v})) - \prod_{\mu\neq u,v}\cos(2\gamma(t)(c'_{\mu u}-c'_{\mu v}))\right),\end{aligned} \tag{12}$$

where $t = \gamma T$ is the total amount of time that the system evolves under the H_{Ising} pulses during execution of the algorithm, γ and β are variational parameters of the algorithm, and T (eqn. (5)) is the amount of time it takes to compile the unitary $\exp(-iC')$. In the noiseless limit $\Gamma \to 0$, the expression (12) agrees with the generic QAOA expectation value in eqn. (14) of Ref. [44], while the value is exponentially suppressed when $\Gamma > 0$. We focus here on single-layer QAOA because we are

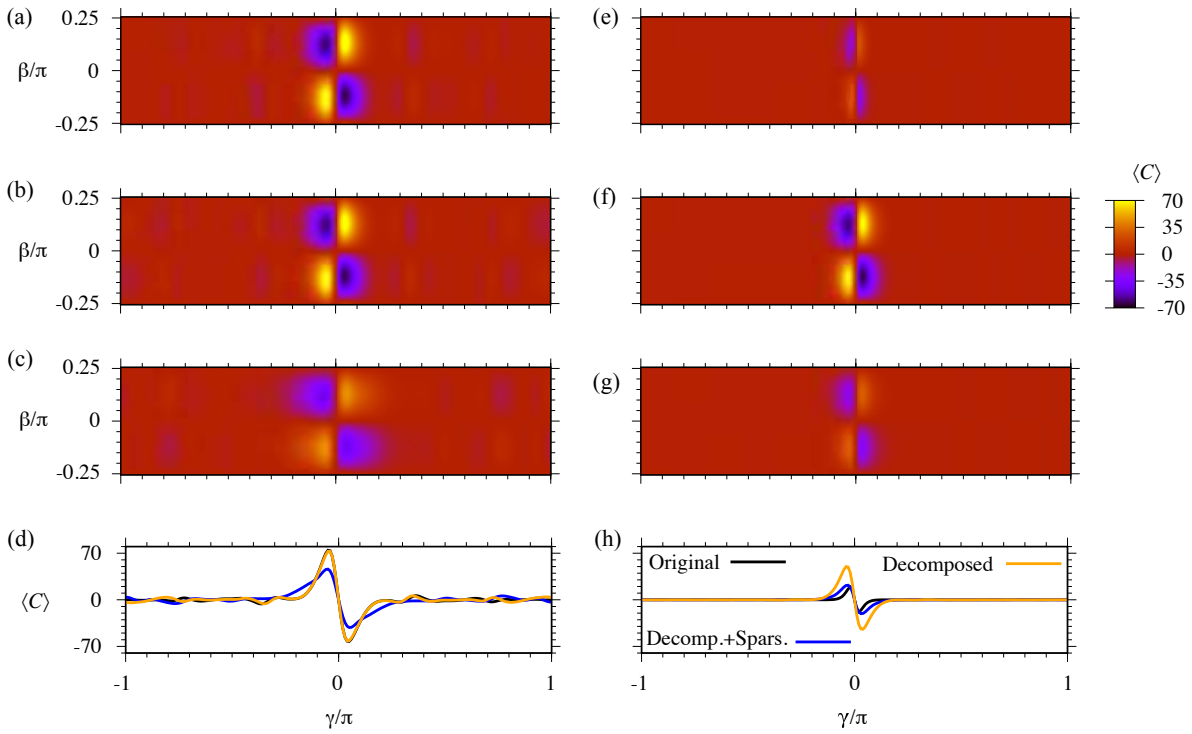


Figure 8: Landscapes of the QAOA cost function. Left column shows the noiseless case $\Gamma = 0$ for (a) standard compilation, (b) compilation with decomposition, (c) compilation with decomposition and sparsification, and (d) line traces through the optimal β for each contour in (a)-(c). The right column (e)-(h) shows analogous results in a noisy case with $\Gamma = 0.001$, and have the same vertical axes with the figures in the left column. The instance shows the median improvement to $\langle C\rangle_{\text{sparsified+decomposed}}/\langle C\rangle_{\text{original}} = 1.1$, and since the optimal $\langle C\rangle_{\text{original}} < 0$, this indicates that $\langle C\rangle_{\text{sparsified+decomposed}} < \langle C\rangle_{\text{original}}$ and therefore the sparsified and decomposed compilation outperforms the original one in minimizing the expected cost (recall eqn. (3) and subsequent discussion); the compilation with decomposition alone achieves an even greater benefit. The graph identifier for this instance is g001098 with sparsification parameter $q = 0.5m$ and $\epsilon_2 = 2$, where m is the number of edges in the graph.

able to evaluate its performance at large sizes using the analytical formula (12). Greater numbers of QAOA layers would be expected to yield higher performance in the noiseless limit, but closed-form expressions for generic QAOA instances at greater numbers of layers have not been presented in the literature to our knowledge, and we have not derived such formulas in the noisy cases analyzed here. Noiseless analysis of QAOA at larger depths is possible in certain highly symmetric instances [3, 19], but extending our noisy analysis to these cases is beyond the scope of this work.

Fig. 8 shows an example of the cost landscape $\langle C\rangle$ as a function of the QAOA parameters γ and β, for cost functions that have been normalized to set the average edge weight to unity following Ref. [54]. The left column shows noiseless cases of (a) the original compilation [50], (b) decomposed compilation, and (c) decomposed and sparsified compilation, with (d) showing a line cut through the optimal β. The results with decomposition are essentially identical to the original compilation, while a much larger error is incurred by the sparsified compilation because the approximate cuts in C' produce an approximate quantum state from the circuit. (e), (f), (g), and (h) show analogous results for the same instance in the presence of noise, with $\Gamma = 0.001$. Here the decomposed compilation greatly outperforms the original compilation, while the sparsified and decomposed compilation slightly outperforms it.

One surprising feature of these results is the extent to which sparsification is harming the performance. In the noiseless case, the lower approximation ratio in Fig. 8 is due to the approximate

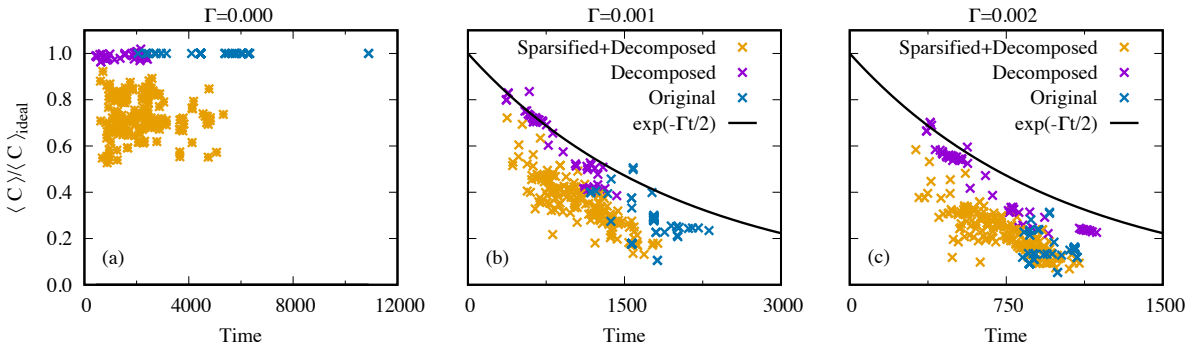


Figure 9: Expected cost $\langle C \rangle$ in QAOA for original, decomposed, and decomposed and sparsified and sparsified compilations, relative to the "ideal" case without noise and with the original compilation. (a) noiseless case, (b) noisy case with $\Gamma = 0.001$, and (c) with $\Gamma = 0.002$. The black curve $\exp(-\Gamma t/2)$ is an upper bound in the triangle-free case as described in the text. Each instance is compiled into three distinct sparsified compilations. All panels share a common vertical axis.

graph coupling C' as mentioned previously. In the noisy cases, there is still little or no benefit in our calculations because sparsification is not correlated with reduced execution time t as seen previously in Fig. 5, while noise in our model depends only on the execution time in eqn. (12). However, it is important to keep in mind that a more realistic noise model may see benefits from sparsification which are not evident in the simple model we consider here. For example, if there is some residual entanglement between the vibrational and electronic modes after each application of H, as described above and as expected in more realistic treatments of trapped-ion dynamics [31], then we would expect that limiting the number of H_{Ising} applications through sparsification (Fig. 3) would produce a less noisy final result. Similar considerations also apply if there is noise associated with bit flip operations (Fig. 4). This gives some reason to expect that sparsification may still be useful for limiting noise in real-world trapped ion experiments. Finally, it is important to note that the analog trapped-ion noise model we consider in this section is distinct from the digital quantum-computing considerations from the previous section, where there is a direct link between sparsification, the number of edges, and the expected reduction in noise.

Next, we examine the optimized performance across a wide variety of instances using the original compilation, decomposition, and for three implementations of sparsification + decomposition for each instance. The standard procedure for implementing QAOA includes optimizing the parameters $\boldsymbol{\gamma}$, $\boldsymbol{\beta}$ for high performance [18]. Many methods for accomplishing this have been studied [54, 32, 35, 69, 19, 1, 68]. Among these grid searches [18, 36] are simple options that are guaranteed to find optimal parameters within the relevant parameter subspace and to within the resolution of the grid spacing. We use grid searches here to obtain these optimal parameters, so there is no uncertainty in the results due to the parameter optimization procedure, allowing us to directly assess the best possible performance, within a reasonable parameter interval, for the various decomposition methods. We identify optimized QAOA parameters using a grid search with resolution of 0.01π in γ and β, over the parameter regions shown in Fig. 8 where high-quality results are expected [54].

In Fig. 9 we show the optimized performance in (a) the noiseless case, (b) a noisy case with $\Gamma = 0.001$, and (c) with $\Gamma = 0.002$. The decomposed compilation is comparable to the original compilation in the noiseless case and significantly outperforms it in the noisy case, while sparsification together with decomposition yields more modest improvements, all as expected from our previous analysis of Fig. 8. To gain a better understanding of the results in Fig. 9, next we will analyze the expected noisy cost under a simplifying assumption, which will explain the approximate upper bound $\exp(-\Gamma t/2)$ pictured in black.

The expected cost eqn. (12) can be expressed as $\langle C \rangle = e^{-\Gamma t/2} f(\gamma, \beta) + e^{-\Gamma t} \tau(\gamma, \beta)$, where the function f [given by the first set of sums in eqn. (12)] describes contributions from each edge (u, v)

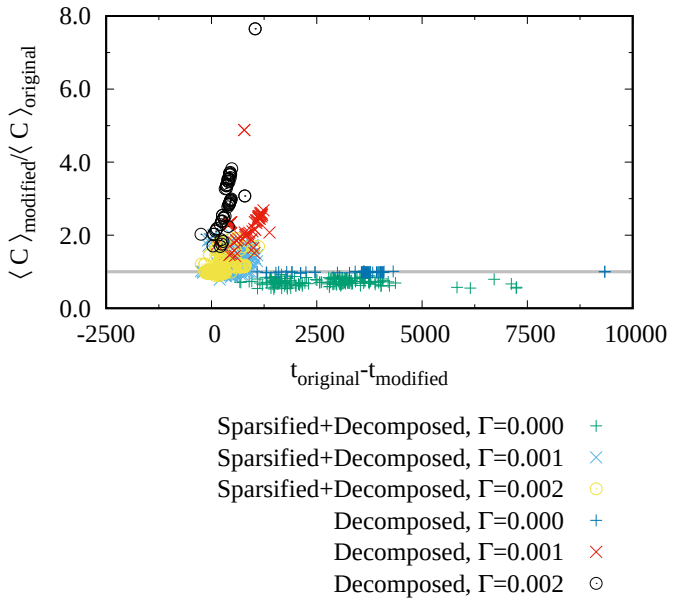


Figure 10: Expected improvement in cost from modified compilations relative to the original compilation. The grey line shows $\langle C\rangle_{\text{modifed}}/\langle C\rangle_{\text{original}} = 1$.

in the graph while the function τ [given by the second set of sums in eqn. (12)] describes additional contributions when there are triangles in the graph. We find that for each of our compilation approaches, the optimized contribution of the triangle term is typically $|\tau(\gamma^*,\beta^*)| < |f(\gamma^*,\beta^*)|/10$. We can therefore approximate $\langle C\rangle \approx e^{-\Gamma t/2} f(\gamma^*,\beta^*)$. We would now like to consider noisy behavior relative to noiseless behavior, starting with the original compilation only. We then have $\langle C\rangle_{\text{noise}}/\langle C\rangle_{\text{ideal}} \approx e^{-\Gamma t'^*/2} f(\gamma'^*,\beta'^*)/f(\gamma^*,\beta^*)$, where γ'^*,β'^* are the chosen parameters in the presence of noise. In the simple fixed-parameter case $(\gamma'^*,\beta'^*) = (\gamma^*,\beta^*)$ this reduces to $\langle C\rangle_{\text{noise}}/\langle C\rangle_{\text{no noise}} = e^{-\Gamma t'^*/2}$. However, as we saw previously in Fig. 8, noise tends to suppress the optimal parameter such that $\gamma'^* < \gamma^*$. In this case $\langle C\rangle_{\text{noise}}/\langle C\rangle_{\text{ideal}} = e^{-\Gamma t'^*/2} f(\gamma'^*,\beta'^*)/f(\gamma^*,\beta^*) < e^{-\Gamma t'^*/2}$, since $f(\gamma'^*,\beta'^*)/f(\gamma^*,\beta^*) < 1$ as γ'^* is not the ideal parameter γ^* that maximizes f. Note that $t'^* \propto \gamma'^*$ and is smaller in this second case than in the $\gamma'^* = \gamma^*$ case. Based on this simple model, we expect that with the original compilation, $\langle C\rangle_{\text{noise}}/\langle C\rangle_{\text{ideal}} \lesssim e^{-\Gamma t/2}$ for an optimized noisy runtime t, with a strict inequality $\langle C\rangle_{\text{noise}}/\langle C\rangle_{\text{ideal}} \leq e^{-\Gamma t/2}$ in the triangle-free case. When we use other compilations, then $\langle C\rangle_{\text{noise}}$ may decrease further due to the use of approximate compilation. So in all cases we expect $\langle C\rangle_{\text{noise}}/\langle C\rangle_{\text{ideal}} \lesssim e^{-\Gamma t/2}$. This simple model gives a good account of the typical behavior observed in Fig. 9.

Finally, in Fig. 10 we give a direct comparison of the costs from our various compilations relative to the original compilation. In the presence of noise, our new compilations consistently outperform the original compilation. This result provides strong evidence that our new compilation approaches here provide superior performance to the original compilation when noise is considered.

5 Conclusion

Noise in quantum hardware presents a significant hurdle to achieving scalable quantum computing. We presented a problem-aware approach for noise reduction in QAOA that modifies the problem instance suitably, resulting in shallower circuit depth and therefore less noise. We presented theoretical bounds and numerical evidence for reduction in the number of circuit gates while guaranteeing high Max-Cut

approximations on trapped-ion hardware employing all-to-all Mølmer-Sørensen interactions. We also presented a generic theoretical argument for how these techniques can improve the fidelity of digital quantum computations with superconducting qubits [9], Rydberg atoms [7], trapped-ions employing two-qubit gates [47], or any other digital quantum computing technology with nonnegligible noise described by Krauss operators. This approach reduces circuit noise and allows QAOA to scale for larger graphs on NISQ hardware, thus narrowing the performance gap between classical and quantum hardware. However, quantum noise remains a major challenge, and classical methods continue to outperform quantum algorithms for combinatorial optimization.

Other approaches to noise reduction in QAOA have been developed in parallel to our work. These include finding smaller graphs with similar energy landscapes [65], integer programming-based approaches for crosstalk noise reduction [65], and using machine learning approaches to predict QAOA parameters [52]. To the best of our knowledge, ours is the only approach with formal worst-case guarantees on the approximation ratio.

While most of our asymptotic theoretical guarantees are restricted to trapped-ion hardware, both sparsification and decomposition are general tools that may be useful beyond the trapped-ion setting. Sparsification is expected to improve noise resilience in generic compilations since it simplifies the problem instance, resulting in fewer gates and an exponentially improved fidelity lower bound as described here. Further, since there exist faster classical combinatorial algorithms with better guarantees for unweighted instances, relative to weighted instances [12, 67], we believe decomposition-like techniques might be useful in analyzing quantum optimization algorithms for weighted instances.

6 Author contributions

This work sits at the intersection of classical and quantum computing: a central challenge was to identify quantum noise models that admit rigorous analysis alongside classical graph sparsification and decomposition techniques. All authors contributed to the design of the study and experiments. Swati Gupta (SG) and Jai Moondra (JM) introduced the idea of classical graph pre-processing to generate QAOA compilations with smaller circuits. Phillip C. Lotshaw (PCL) and Greg Mohler (GM) introduced the quantum noise models, with PCL leading the noise analysis and the derivation of the dephased QAOA cost expectation. JM led the analysis of the sparsification and decomposition algorithms and the resulting reductions in gate counts; the theoretical guarantees were developed jointly by JM and SG with input from all authors. PCL ran all quantum simulations; JM performed the classical sparsification and decomposition experiments, both with input from all authors. All authors contributed to writing and editing the manuscript.

Acknowledgements. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0046. The authors would like to thank Brandon Augustino, Cameron Dahan, Bryan Gard, Mehrdad Ghadiri, Creston Herold, Sarah Powers, and Mohit Singh for their careful comments and helpful discussions on this work.

Notes

[1]We borrow this notation from algorithmic graph theory and say an event occurs 'with high probability' if the corresponding probability p_n goes to 1 as the number n of vertices of the graph goes to infinity, i.e., $p_n = 1 - o(1)$.

[2]Effective resistance of an edge in a graph is the equivalent resistance measured between its endpoints, treating the edge weights as inverse resistances. For details on computing this quantity, we refer the interested reader to [58].

References

[1] B. Augustino, M. Cain, E. Farhi, S. Gupta, S. Gutmann, D. Ranard, E. Tang, and K. Van Kirk. Strategies for running the QAOA at hundreds of qubits. *arXiv:2410.03015*, 2024. DOI: https://doi.org/10.48550/arXiv.2410.03015.

[2] B. Augustino, G. Nannicini, T. Terlaky, and L. Zuluaga. Quantum interior point methods for semidefinite optimization. *Quantum*, 7:1110, 2023. DOI: https://doi.org/10.22331/q-2023-09-11-1110.

[3] J. Basso, E. Farhi, K. Marwaha, B. Villalonga, and L. Zhou. The quantum approximate optimization algorithm at high depth for MaxCut on large-girth regular graphs and the Sherrington-Kirkpatrick model. In *17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC)*, 2022. DOI: https://doi.org/10.4230/LIPIcs.TQC.2022.7.

[4] J. Batson, D. Spielman, and N. Srivastava. Twice-Ramanujan sparsifiers. *SIAM Review*, 56(2):315–334, 2014. DOI: https://doi.org/10.1137/090772873.

[5] L. Binkowski, G. Koßmann, T. Ziegler, and R. Schwonnek. Elementary proof of QAOA convergence. *New Journal of Physics*, 26(7):073001, 2024. DOI: https://doi.org/10.1088/1367-2630/ad59bb.

[6] R. Blümel, N. Grzesiak, N. Pisenti, K. Wright, and Y. Nam. Power-optimal, stabilized entangling gate between trapped-ion qubits. *npj Quantum Information*, 7(1):147, 2021. DOI: https://doi.org/10.1038/s41534-021-00489-w.

[7] D. Bluvstein, S. Evered, A. Geim, S. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, et al. Logical quantum processor based on reconfigurable atom arrays. *Nature*, 626(7997):58–65, 2024. DOI: https://doi.org/10.1038/s41586-023-06927-3.

[8] J. Bohnet, B. Sawyer, J. Britton, M. Wall, A. Rey, M. Foss-Feig, and J. Bollinger. Quantum spin dynamics and entanglement generation with hundreds of trapped ions. *Science*, 352(6291):1297–1301, 2016. DOI: https://doi.org/10.1126/science.aad9958.

[9] S. Bravyi, O. Dial, J. Gambetta, D. Gil, and Z. Nazario. The future of quantum computing with superconducting qubits. *Journal of Applied Physics*, 132(16), 2022. DOI: https://doi.org/10.1063/5.0082975.

[10] M. Charikar, T. Leighton, S. Li, and A. Moitra. Vertex sparsifiers and abstract rounding algorithms. In *51st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 265–274, 2010. DOI: https://doi.org/10.1109/FOCS.2010.32.

[11] C. Clark, H. Tinkey, B. Sawyer, A. Meier, K. Burkhardt, C. Seck, C. Shappert, N. Guise, C. Volin, S. Fallek, H. Hayden, W. Rellergert, and K. Brown. High-fidelity Bell-state preparation with $^{40}Ca^{+}$ optical qubits. *Physical Review Letters*, 127(13):130505, 2021. DOI: https://doi.org/10.1103/PhysRevLett.127.130505.

[12] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. 2022.

[13] D. Liu, J. Li, X. Chen, and N. Jiang. Noisy quantum approximation optimization algorithm for solving MaxCut problem. *Physica Scripta*, 2025. DOI: https://doi.org/10.1088/1402-4896/ae009f.

[14] I. Dunning, S. Gupta, and J. Silberholz. What works best when? A systematic evaluation of heuristics for Max-Cut and QUBO. *INFORMS Journal on Computing*, 30(3):608–624, 2018. DOI: https://doi.org/10.1287/ijoc.2017.0798.

[15] S. Ebadi, A. Keesling, M. Cain, T. Wang, H. Levine, D. Bluvstein, G. Semeghini, A. Omran, J.-G. Liu, R. Samajdar, et al. Quantum optimization of maximum independent set using Rydberg atom arrays. *Science*, 376(6598):1209–1215, 2022. DOI: https://doi.org/10.1126/science.abo6587.

[16] D. Egger, J. Mareček, and S. Woerner. Warm-starting quantum optimization. *Quantum*, 5:479, 2021. DOI: https://doi.org/10.22331/q-2021-06-17-479.

[17] E. Farhi, D. Gamarnik, and S. Gutmann. The quantum approximate optimization algorithm needs to see the whole graph: A typical case. *arXiv:2004.09002*, 2020. DOI: https://doi.org/10.48550/arXiv.2004.09002.

[18] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm. *arXiv:1411.4028*, 2014. DOI: https://doi.org/10.48550/arXiv.1411.4028.

[19] E. Farhi, J. Goldstone, S. Gutmann, and L. Zhou. The quantum approximate optimization algorithm and the Sherrington-Kirkpatrick model at infinite size. *Quantum*, 6:759, 2022. DOI: https://doi.org/10.22331/q-2022-07-07-759.

[20] M. Foss-Feig, K. Hazzard, J. Bollinger, and A. Rey. Nonequilibrium dynamics of arbitrary-range Ising models with decoherence: An exact analytic solution. *Physical Review A*, 87(4):042101, 2013. DOI: https://doi.org/10.1103/PhysRevA.87.042101.

[21] M. Goemans and D. Williamson. .879-approximation algorithms for MAX CUT and MAX 2SAT. In *26th ACM Symposium on Theory of Computing (STOC)*, pages 422–431, 1994. DOI: https://doi.org/10.1145/195058.195216.

[22] L. Grover. A fast quantum mechanical algorithm for database search. In *28th ACM Symposium on Theory of Computing (STOC)*, pages 212–219, 1996. DOI: https://doi.org/10.1145/237814.237866.

[23] S. Harwood, C. Gambella, D. Trenev, A. Simonetto, D. Bernal, and D. Greenberg. Formulating and solving routing problems on quantum computers. *IEEE Transactions on Quantum Engineering*, 2:1–17, 2021. DOI: https://doi.org/10.1109/TQE.2021.3049230.

[24] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001. DOI: https://doi.org/10.1145/502090.502098.

[25] D. Karger. Random sampling in cut, flow, and network design problems. In *26th ACM Symposium on Theory of Computing (STOC)*, pages 648–657, 1994. DOI: https://doi.org/10.1287/moor.24.2.383.

[26] H. Karloff. How good is the Goemans–Williamson MAX CUT algorithm? *SIAM Journal on Computing*, 29(1):336–350, 1999. DOI: https://doi.org/10.1137/S0097539797321481.

[27] I. Kerenidis and A. Prakash. A quantum interior point method for LPs and SDPs. *ACM Transactions on Quantum Computing*, 1(1):5:1–5:32, 2020. DOI: https://doi.org/10.1145/3406306.

[28] S. Khot. On the power of unique 2-prover 1-round games. In *34th ACM Symposium on Theory of Computing (STOC)*, pages 767–775, 2002. DOI: https://doi.org/10.1109/CCC.2002.1004334.

[29] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann, G. Norris, C. Andersen, M. Müller, A. Blais, C. Eichler, and A. Wallraff. Realizing repeated quantum error correction in a distance-three surface code. *Nature*, 605(7911):669–674, 2022. DOI: https://doi.org/10.1038/s41586-022-04566-8.

[30] X. Liu, R. Shaydulin, and I. Safro. Quantum approximate optimization algorithm with sparsified phase operator. In *IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 133–141, 2022. DOI: https://doi.org/10.1109/QCE53715.2022.00032.

[31] P. Lotshaw, K. Battles, B. Gard, G. Buchs, T. Humble, and C. Herold. Modeling noise in global Mølmer-Sørensen interactions applied to quantum approximate optimization. *Physical Review A*, 107(6):062406, 2023. DOI: https://doi.org/10.1103/PhysRevA.107.062406.

[32] P. Lotshaw, T. Humble, R. Herrman, J. Ostrowski, and G. Siopsis. Empirical performance bounds for quantum approximate optimization. *Quantum Information Processing*, 20(12):403, 2021. DOI: https://doi.org/10.1007/s11128-021-03342-3.

[33] P. Lotshaw, T. Nguyen, A. Santana, A. McCaskey, R. Herrman, J. Ostrowski, G. Siopsis, and T. Humble. Scaling quantum approximate optimization on near-term hardware. *Scientific Reports*, 12(1):12388, 2022. DOI: https://doi.org/10.1038/s41598-022-14767-w.

[34] P. Lotshaw, B. Sawyer, C. Herold, and G. Buchs. Exactly solvable model of light-scattering errors in quantum simulations with metastable trapped-ion qubits. *Physical Review A*, 110(3):L030803, 2024. DOI: https://doi.org/10.1103/PhysRevA.110.L030803.

[35] P. Lotshaw, G. Siopsis, J. Ostrowski, R. Herrman, R. Alam, S. Powers, and T. Humble. Approximate Boltzmann distributions in quantum approximate optimization. *Physical Review A*, 108(4):042411, 2023. DOI: https://doi.org/10.1103/PhysRevA.108.042411.

[36] P. Lotshaw, H. Xu, B. Khalid, G. Buchs, T. Humble, and A. Banerjee. Simulations of frustrated Ising Hamiltonians using quantum approximate optimization. *Philosophical Transactions of the Royal Society A*, 381(2241):20210414, 2023. DOI: https://doi.org/10.1098/rsta.2021.0414.

[37] A. Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2:74887, 2014. DOI: https://doi.org/10.3389/fphy.2014.00005.

[38] A. Moitra. Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size. In *50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 3–12, 2009. DOI: https://doi.org/10.1109/FOCS.2009.28.

[39] C. Monroe, W. Campbell, L.-M. Duan, Z.-X. Gong, A. Gorshkov, P. Hess, R. Islam, K. Kim, N. Linke, G. Pagano, et al. Programmable quantum simulations of spin systems with trapped ions. *Reviews of Modern Physics*, 93(2):025001, 2021. DOI: https://doi.org/10.1103/RevModPhys.93.025001.

[40] J. Moondra, P. Lotshaw, G. Mohler, and S. Gupta. Source code for "Promise of graph sparsification and decomposition for noise reduction in QAOA: Analysis for trapped-ion compilations", 2026. DOI: https://doi.org/10.5281/zenodo.20669870

[41] T. Morris and P. Lotshaw. Performant near-term quantum combinatorial optimization. *arXiv:2404.16135*, 2024. DOI: https://doi.org/10.48550/arXiv.2404.16135.

[42] G. Nannicini. Fast quantum subroutines for the simplex method. *Operations Research*, 72(2):763–780, 2024. DOI: https://doi.org/10.1287/opre.2022.2341.

[43] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. 2001.

[44] A. Ozaeta, W. van Dam, and P. McMahon. Expectation values from the single-layer quantum approximate optimization algorithm on Ising problems. *Quantum Science and Technology*, 7(4):045036, 2022. DOI: https://doi.org/10.1088/2058-9565/ac9013.

[45] G. Pagano, A. Bapat, P. Becker, K. Collins, A. De, P. Hess, H. Kaplan, A. Kyprianidis, W. Tan, C. Baldwin, L. Brady, A. Deshpande, F. Liu, S. Jordan, A. Gorshkov, and C. Monroe. Quantum approximate optimization with a trapped-ion quantum simulator. 2019. DOI: https://doi.org/10.48550/arXiv.1906.02700.

[46] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. Love, A. Aspuru-Guzik, and J. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1):4213, 2014. DOI: https://doi.org/10.1038/ncomms5213.

[47] J. Pino, J. Dreiling, C. Figgatt, J. Gaebler, S. Moses, M. Allman, C. Baldwin, M. Foss-Feig, D. Hayes, K. Mayer, et al. Demonstration of the trapped-ion quantum CCD computer architecture. *Nature*, 592(7853):209–213, 2021. DOI: https://doi.org/10.1038/s41586-021-03318-4.

[48] M. Plenio and P. Knight. The quantum-jump approach to dissipative dynamics in quantum optics. *Reviews of Modern Physics*, 70(1):101, 1998. DOI: https://doi.org/10.1103/RevModPhys.70.101.

[49] J. Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2:79, 2018. DOI: https://doi.org/10.22331/q-2018-08-06-79.

[50] J. Rajakumar, J. Moondra, B. Gard, S. Gupta, and C. Herold. Generating target graph couplings for the quantum approximate optimization algorithm from native quantum hardware couplings. *Physical Review A*, 106(2):022606, 2022. DOI: https://doi.org/10.1103/PhysRevA.106.022606.

[51] C. Ryan-Anderson, J. Bohnet, K. Lee, D. Gresh, A. Hankin, J. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. Brown, T. Gatterman, S. Halit, K. Gilmore, J. Gerber, B. Neyenhuis, D. Hayes, and R. Stutz. Realization of real-time fault-tolerant quantum error correction. *Physical Review X*, 11(4):041058, 2021. DOI: https://doi.org/10.1103/PhysRevX.11.041058.

[52] S. Sack. Large-scale quantum approximate optimization on nonplanar graphs with machine learning noise mitigation. *Physical Review Research*, 6(1), 2024. DOI: https://doi.org/10.1103/PhysRevResearch.6.013223.

[53] R. Shaydulin, C. Li, S. Chakrabarti, M. DeCross, D. Herman, N. Kumar, J. Larson, D. Lykov, P. Minssen, Y. Sun, et al. Evidence of scaling advantage for the quantum approximate optimization

algorithm on a classically intractable problem. *Science Advances*, 10(22):eadm6761, 2024. DOI: https://doi.org/10.1126/sciadv.adm6761.

[54] R. Shaydulin, P. Lotshaw, J. Larson, J. Ostrowski, and T. Humble. Parameter transfer for quantum approximate optimization of weighted MaxCut. *ACM Transactions on Quantum Computing*, 4(3):1–15, 2023. DOI: https://doi.org/10.1145/3584706.

[55] R. Shaydulin, I. Safro, and J. Larson. Multistart methods for quantum approximate optimization. In *IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–8, 2019. DOI: https://doi.org/10.1109/HPEC.2019.8916288.

[56] P. Shor. Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, 52(4):R2493–R2496, 1995. DOI: https://doi.org/10.1103/PhysRevA.52.R2493.

[57] V. Sivak, A. Eickbusch, B. Royer, S. Singh, I. Tsioutsios, S. Ganjam, A. Miano, B. Brock, A. Ding, L. Frunzio, S. Girvin, R. Schoelkopf, and M. Devoret. Real-time quantum error correction beyond break-even. *Nature*, 616(7955):50–55, 2023. DOI: https://doi.org/10.1038/s41586-023-05782-6.

[58] D. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011. DOI: https://doi.org/10.1137/080734029.

[59] D. Sutter, G. Nannicini, T. Sutter, and S. Woerner. Quantum speedups for convex dynamic programming. *arXiv:2011.11654*, 2021. DOI: https://doi.org/10.48550/arXiv.2011.11654.

[60] B. Tasseff, T. Albash, Z. Morrell, M. Vuffray, A. Lokhov, S. Misra, and C. Coffrin. On the emerging potential of quantum annealing hardware for combinatorial optimization. *arXiv:2210.04291*, 2022. DOI: https://doi.org/10.48550/arXiv.2210.04291.

[61] R. Tate, J. Moondra, B. Gard, G. Mohler, and S. Gupta. Warm-started QAOA with custom mixers provably converges and computationally beats Goemans–Williamson's Max-Cut at low circuit depths. *Quantum*, 7:1121, 2023. DOI: https://doi.org/10.22331/q-2023-09-26-1121.

[62] H. Uys, M. Biercuk, A. VanDevender, C. Ospelkaus, D. Meiser, R. Ozeri, and J. Bollinger. Decoherence due to elastic Rayleigh scattering. *Physical Review Letters*, 105(20):200401, 2010. DOI: https://doi.org/10.1103/PhysRevLett.105.200401.

[63] C. Valahu, I. Apostolatos, S. Weidt, and W. Hensinger. Quantum control methods for robust entanglement of trapped ions. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 55(20):204003, 2022. DOI: https://doi.org/10.1088/1361-6455/ac8eff.

[64] V. Vazirani. *Approximation Algorithms*. 2003.

[65] M. Wang, B. Fang, A. Li, and P. Nair. Red-QAOA: Efficient variational optimization through circuit reduction. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 980–998, 2024. DOI: https://doi.org/10.1145/3620665.3640363.

[66] D. West. *Introduction to Graph Theory*. 2001.

[67] D. Williamson and D. Shmoys. *The Design of Approximation Algorithms*. 2010.

[68] J. Wurtz and D. Lykov. Fixed-angle conjectures for the quantum approximate optimization algorithm on regular MaxCut graphs. *Physical Review A*, 104(5):052419, 2021. DOI: https://doi.org/10.1103/PhysRevA.104.052419.

[69] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Physical Review X*, 10(2):021067, 2020. DOI: https://doi.org/10.1103/PhysRevX.10.021067.

A Omitted proofs from Section 3

We analyze our algorithms for faster UNION-OF-STARS from Section 3 and prove their theoretical guarantees.

A.1 Analysis of Algorithm EXP-DECOMPOSE

Proof of Theorem 3. Recall that in Algorithm EXP-DECOMPOSE , we set $c^* = \max_{e\in E} c(e)$ and threshold $\tau = \frac{\epsilon c^*}{2n^2}$. Correspondingly we set $k = \lceil \log_{1+\epsilon/2} \tau \rceil = O\left(\frac{\log(n/\epsilon)}{\epsilon}\right)$.

Consider a cut $S \subseteq V$. By construction in the algorithm, $c'(e) \le c(e)$ for all edges $e \in E$, and therefore the cut value can only decrease in $G' = (V, E', c')$, i.e., $\delta_{G'}(S) \le \delta_G(S)$. We next show that it does not decrease by too much, i.e., $\delta_{G'}(S) \ge (1-\epsilon)\delta_G(S)$ if S is a non-trivial cut in G, i.e., if $\delta_G(S) \ge \frac{1}{2}\sum_{e\in E} c(e)$.

Fix edge $e \in E$. Denote $E_{\text{small}} := \{e \in E : c(e) < \tau\}$ and $E_{\text{large}} := \{e \in E : c(e) \ge \tau\}$. If $e \in E_{\text{small}}$, then $e' \notin E'$ by construction in the algorithm. If $e \in E_{\text{large}}$, then by construction in line 7, we get $c'(e) \ge \left(1 - \frac{\epsilon}{2}\right) c(e)$.

Noting that $\Delta_{G'}(S) = \Delta_G(S) \cap E_{\text{large}}$, the decrease in the cut value for S can be bounded as:

$$\begin{aligned}
\delta_G(S) - \delta_{G'}(S) &= \sum_{e\in\Delta_G(S)} c(e) - \sum_{e\in\Delta_{G'}(S)} c'(e) \\
&= \sum_{e\in\Delta_G(S)\cap E_{\text{small}}} c(e) + \sum_{e\in\Delta_G(S)\cap E_{\text{large}}} c(e) - \sum_{e\in\Delta_G(S)\cap E_{\text{large}}} c'(e) \\
&\le \tau|\Delta_G(S)\cap E_{\text{small}}| + \frac{\epsilon}{2}\sum_{e\in\Delta_G(S)\cap E_{\text{large}}} c(e).
\end{aligned}$$

We will claim that each of the terms above is at most $\frac{\epsilon}{2}\delta_G(S)$, which is sufficient to prove part (1) of the theorem.

It is easy to see that $\frac{\epsilon}{2}\sum_{e\in\Delta_G(S)\cap E_{\text{large}}} c(e) \le \frac{\epsilon}{2}\sum_{e\in\Delta_G(S)} c(e) = \frac{\epsilon}{2}\delta_G(S)$. For the first term, plugging in $\tau = \frac{\epsilon c^*}{2n^2}$,

$$\begin{aligned}
\tau|\Delta_G(S)\cap E_{\text{small}}| &= \frac{\epsilon c^*}{2n^2}|\Delta_G(S)\cap E_{\text{small}}| \\
\le \frac{\epsilon c^*}{2n^2}|E| \le \frac{\epsilon c^*}{2n^2}\cdot\frac{n^2}{2} &\le \frac{\epsilon}{4}c^*.
\end{aligned}$$

However, $c^* \le \sum_{e\in E} c(e) \le 2\delta_G(S)$, since S is a non-trivial cut by assumption. Therefore, $\frac{\epsilon}{4}c^* \le \frac{\epsilon}{2}\delta_G(S)$. This completes the proof of the claim.

To see part (2), note that by construction, E' is the disjoint union of $\bigcup_{j\in[k]} E_j$. Further, each of $G = (V, E_j)$ is an unweighted graph. By Lemma 2 and Lemma 1, we have that

$$\begin{aligned}
&N(H_{\text{ISING}})(G') \le k\cdot(3n-2) \\
&= \left(\log_{1+\frac{\epsilon}{2}}\frac{2n^2}{\epsilon}\right)\cdot(3n-2) = O\left(\frac{n}{\epsilon}\log\frac{n}{\epsilon}\right).
\end{aligned}$$

□

A.2 Analysis of Algorithm BINARY-DECOMPOSE

Proof of Theorem 4. Denote $G = (V, E, c)$. Let c^*, η, k be as in Algorithm BINARY-DECOMPOSE . Recall that for each $e \in E$, we define $d(e) = \lfloor \frac{c(e)}{\eta} \rfloor$ for each $e \in E$, and $b_{k-1}(e)b_{k-2}(e)\dots b_0(e)$ are the digits in the binary representation of $d(e)$. Therefore, $d(e) = \sum_{j\in[k]} 2^{j-1}b_{j-1}(e)$.

Let $G_1, \dots, G_k$ be the unweighted graphs in Algorithm BINARY-DECOMPOSE . Recall that the algorithm returns $G' = (V, E', c')$ such that $G' = \sum_{j\in[k]} \eta 2^{j-1} G_j$.

(i) For each edge $e \in E$, the edge weight in G' is $c'(e) = \sum_{j\in[k]} \eta 2^{j-1} b_{j-1}(e) = \eta d(e)$. Then we get

$$c'(e) = \eta d(e) = \eta \left\lfloor \frac{c(e)}{\eta} \right\rfloor \geq \eta \left(\frac{c(e)}{\eta} - 1 \right) = c(e) - \eta.$$

Also, $c'(e) \leq c(e)$. Let $S \subseteq V$ be any set of vertices; there are at most $n^2/2$ edges in the corresponding cut. Summing across all edges in the cut, we get

$$\delta_G(S) - \delta_{G'}(S) \leq |E|\eta = |E|\frac{c^*\epsilon}{n^2} \leq \frac{c^*}{2}\epsilon \leq \left(\frac{1}{2} \sum_{e\in E} c(e) \right) \epsilon.$$

Since S is a non-trivial cut, $\delta_G(S) \geq \frac{1}{2}\sum_{e\in E} c(e)$, so that $\delta_G(S) - \delta_{G'}(S) \leq \epsilon\delta_G(S)$. That is, $\delta_{G'}(S) \geq (1-\epsilon)\delta_G(S)$.

In particular, if $\hat{S}, \hat{S}'$ are the vertex sets corresponding to Max-Cut in G, G' respectively, we have

$$\delta_G(\hat{S}') \geq \delta_{G'}(\hat{S}') \geq \delta_{G'}(\hat{S}) \geq (1-\epsilon)\delta_G(\hat{S}).$$

The first inequality holds since $c'(e) \leq c(e)$ for all $e \in E$, the second inequality holds since $\hat{S}$ is the Max-Cut in G', and the third inequality holds since $\hat{S}$ is a non-trivial cut. Therefore, the Max-Cut $\hat{S}'$ in G' is a $(1-\epsilon)$-approximate cut in G.

(ii) Since each $G_j, j \in [k]$ is an unweighted graph, we have $N(H_{\text{ISING}})(G_j) \leq 3n-2$ from Theorem 1. Since $G' = \sum_{j\in[k]} \eta 2^{j-1} G_j$, we get from Lemma 2 that

$$\begin{aligned} N(H_{\text{ISING}})(G) &\leq \sum_{j\in[k]} N(H_{\text{ISING}})(G_j) \\ &\leq k(3n-2) = O(n\log(n/\epsilon)). \end{aligned}$$

Finally, note that each edge in G' is an edge in at least one of $G_1, \ldots, G_k$, and therefore it is an edge in G. □

A.3 Analysis of Algorithm SPARSE-UNION-OF-STARS

Proof of Theorem 5. Let $H = (V, \overline{E}, \overline{c})$ be the sparsified graph in line 1 of Algorithm SPARSE-UNION-OF-STARS . Then by Theorem 2, with high probability, for all $S \subseteq V$, $1 - \frac{\epsilon}{3} \leq \frac{\delta_H(S)}{\delta_G(S)} \leq 1 + \frac{\epsilon}{3}$. Further, $|\overline{E}| = O\left(\frac{n}{\epsilon^2}\right)$.

Case I. $G' = G'_{\text{bin}} = \text{BINARY-DECOMPOSE}(H, \epsilon_2)$. From Theorem 4, G' satisfies that (1) $N(H_{\text{ISING}})(G') = O\left(n\log(n/\epsilon)\right)$ and (2) $1 - \frac{\epsilon}{3} \leq \frac{\delta_{G'}(S)}{\delta_G(S)} \leq 1$ for all non-trivial cuts $S \subseteq V$. This latter equation, along with the observation above, gives that for all $\epsilon \in (0, 1]$,

$$1 + \frac{\epsilon}{3} \geq \frac{\delta_{G'}(S)}{\delta_G(S)} \geq \left(1 - \frac{\epsilon}{3}\right)^2 = 1 - \frac{2\epsilon}{3} + \frac{\epsilon^2}{9} \geq 1 - \epsilon.$$

Let $G_1, \ldots, G_k$ be the graphs in BINARY-DECOMPOSE for inputs $H, \epsilon/3$. Recall that $k = O(\log(n/\epsilon))$. Since $E(G_j) \subseteq \overline{E}$ for each $j \in [k]$, we get that $N(\text{TOTAL}_{\text{OPS}})(G_j) = O(|E(G_j)|) = O\left(\frac{n}{\epsilon^2}\right)$. Therefore, from Lemma 2,

$$\begin{aligned} N(\text{TOTAL}_{\text{OPS}})(G') &\leq \sum_{j\in[k]} N(\text{TOTAL}_{\text{OPS}})(G_j) \\ &= O\left(\frac{n\log(n/\epsilon)}{\epsilon^2}\right). \end{aligned}$$

Case II. $G' = G'_{\text{exp}} = \text{EXP-DECOMPOSE}(H, \epsilon_2)$. By the choice of G' and from Theorem 3, G' satisfies that $N(H_{\text{ISING}})(G') = N(H_{\text{ISING}})(G'_{\text{exp}}) \leq N(H_{\text{ISING}})(G'_{\text{bin}}) = O\left(n \log(n/\epsilon)\right)$.

Let $G_1, \ldots, G_k$ be the graphs in EXP-DECOMPOSE for inputs $H, \epsilon/3$. Recall that $k = O\left(\frac{1}{\epsilon}\log(\frac{n}{\epsilon})\right)$. Since $E(G') \subseteq E(H)$ is the disjoint union of $E(G_j), j \in [k]$, we get that

$$\sum_j N(\text{TOTAL}_{\text{OPS}})(G_j) = \sum_j O(|E(G_j)|)$$
$$= O(|E(H)|) = O\left(\frac{n}{\epsilon^2}\right).$$

Therefore, from Lemma 2,

$$N(\text{TOTAL}_{\text{OPS}})(G') \leq \sum_{j\in[k]} N(\text{TOTAL}_{\text{OPS}})(G_j) = O\left(\frac{n}{\epsilon^2}\right).$$

Finally, from Theorem 3, $1 - \frac{\epsilon}{3} \leq \frac{\delta_{G'}(S)}{\delta_G(S)} \leq 1$ for all non-trivial cuts S. Therefore, for all such cuts, as before,

$$1 + \frac{\epsilon}{3} \geq \frac{\delta_{G'}(S)}{\delta_G(S)} \geq \left(1 - \frac{\epsilon}{3}\right)^2 = 1 - \frac{2\epsilon}{3} + \frac{\epsilon^2}{9} \geq 1 - \epsilon. \square$$

B A comparison of the two decomposition algorithms

We presented two algorithms for decomposition in Section 3.1: BINARY-DECOMPOSE and EXP-DECOMPOSE. Given decomposition error parameter ϵ_2, both algorithms improve $N(H_{\text{ISING}})$ for weighted graphs from $O(m) = O(n^2)$ to $O\left(\frac{n}{\epsilon_2}\log\frac{n}{\epsilon_2}\right)$ (for EXP-DECOMPOSE) or to $O\left(n\log\frac{n}{\epsilon_2}\right)$ (for BINARY-DECOMPOSE). While the two are similar in terms of dependence on n, the latter depends only logarithmically on $1/\epsilon_2$ while the former depends linearly on $1/\epsilon_2$. Therefore, one would expect that BINARY-DECOMPOSE would outperform EXP-DECOMPOSE.

As we noted in our experiments on MQLib graphs, this is usually *not* the case for smaller graphs and for relatively large ϵ_2. This is because of the large constant for BINARY-DECOMPOSE hidden in the O notation. Indeed, for medium-sized graphs such as the ones we use for our experiments in Section 3.3, using BINARY-DECOMPOSE typically *increases* $N(H_{\text{ISING}})$ and $N(\text{TOTAL}_{\text{OPS}})$ from base UNION-OF-STARS (see Figures 11, 12), as opposed to EXP-DECOMPOSE which significantly decreased these numbers (Figures 3 4).

However, for large and dense enough graphs and for small values of ϵ_2 (i.e., when we seek higher cut quality), BINARY-DECOMPOSE does outperform EXP-DECOMPOSE. We gave example of one such graph from MQLib in Figure 7; Figure 13 presents several more examples. For this plot, we chose the nine densest graphs in MQLib with fewer than 500,000 edges.

C Derivation of the dephased QAOA cost expectation

We model noisy compilation using a dephasing master equation that is a special case of the model presented in Foss-Feig *et. al* [20]. We begin by considering only the native Hamiltonian $H = n^{-1}H_{\text{Ising}}$, then include the σ_u^x from UNION-OF-STARSlater. Define a Lindbladian master equation

$$\frac{d\rho}{dt} = -i(H_{\text{eff}}\rho - \rho H_{\text{eff}}^\dagger) + D(\rho) \tag{13}$$

with an effective non-Hermitian Hamiltonian

$$H_{\text{eff}} = n^{-1}H_{\text{Ising}} - i\sum_{\mathcal{J}} \mathcal{J}^\dagger\mathcal{J} \tag{14}$$

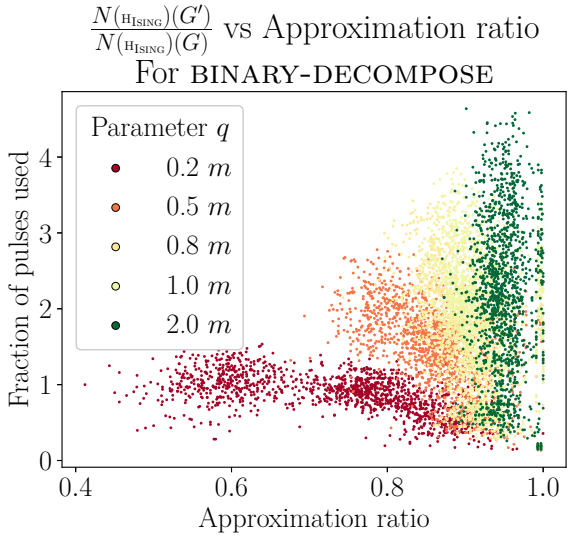


Figure 11: $N(H_{\text{ISING}})(G')/N(H_{\text{ISING}})(G)$ (total number of H_{Ising} pulses) vs Max-Cut approximation for various runs of our experiment on weighted graphs in MQLib using BINARY-DECOMPOSE as the decomposition algorithm, colored by parameter q. Each data point is a single run.

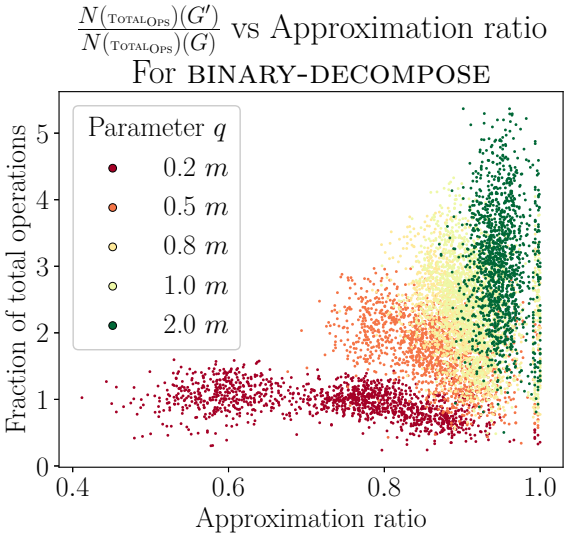


Figure 12: $N(\text{TOTAL}_{\text{OPS}})(G')/N(\text{TOTAL}_{\text{OPS}})(G)$ (total number of H_{Ising} pulses and bit flip gates) vs Max-Cut approximation for various runs of our experiment on weighted graphs in MQLib using BINARY-DECOMPOSE as the decomposition algorithm. Each data point is a single run.

and dissipator

$$\mathcal{D}(\rho) = 2\sum_{\mathcal{J}} \mathcal{J}\rho\mathcal{J}^\dagger \tag{15}$$

which are each specified in terms of a set of jump operators $\mathcal{J}$. Specializing to the case of dephasing only, the set of jump operators is defined as

$$\{\mathcal{J}\} = \left\{\sqrt{\Gamma/8}\sigma_u^z\right\} \tag{16}$$

We consider the evolution of a master equation using the quantum trajectories approach [20, 48]. The evolution is expressed in terms of an ensemble of pure states evolving under the effective Hamiltonian H_{eff}, with jump operators that are randomly interspersed in the evolution, following a probability distribution that causes a large ensemble of randomly generated pure states to exactly reproduce the time-evolving density matrix from (13). The probability of applying a jump operator $\mathcal{J}$ at time t is proportional to $\langle\mathcal{J}^\dagger\mathcal{J}\rangle$, which is simply a constant for the dephasing noise we consider since ${\sigma_u^z}^\dagger\sigma_u^z = 1$. Following Ref. [20], we will compute spin-spin correlations along a single trajectory, then average these over the ensemble of all trajectories to obtain exact correlation functions from the density matrix.

A single trajectory $T = \{\mathcal{J}_{u_1}^{(t_1)}, \mathcal{J}_{u_2}^{(t_2)}, \ldots, \mathcal{J}_{u_K}^{(t_K)}\}$ with jump operators randomly assigned at times $t_1, \ldots, t_K$ will have the form

$$\begin{aligned}|\psi(t;T)\rangle &= e^{-iH_{\text{eff}}(t-t_K)}\mathcal{J}_{u_K}e^{-iH_{\text{eff}}(t_K-t_{K-1})}\mathcal{J}_{u_{K-1}}e^{-iH_{\text{eff}}(t_{K-1}-t_{K-2})}\ldots\mathcal{J}_{u_1}e^{-iH_{\text{eff}}(t_1)}|\psi(0)\rangle \\ &= \mathcal{T}\left(e^{-iH_{\text{eff}}t}\prod_i \mathcal{J}_{u_i}\right)|\psi(0)\rangle\end{aligned} \tag{17}$$

where the last line defines a time-ordering operator $\mathcal{T}$ to abbreviate the notation. If we assume there is no noise associated with bit-flip operations σ_u^x in a UNION-OF-STARS compilation, then we can add

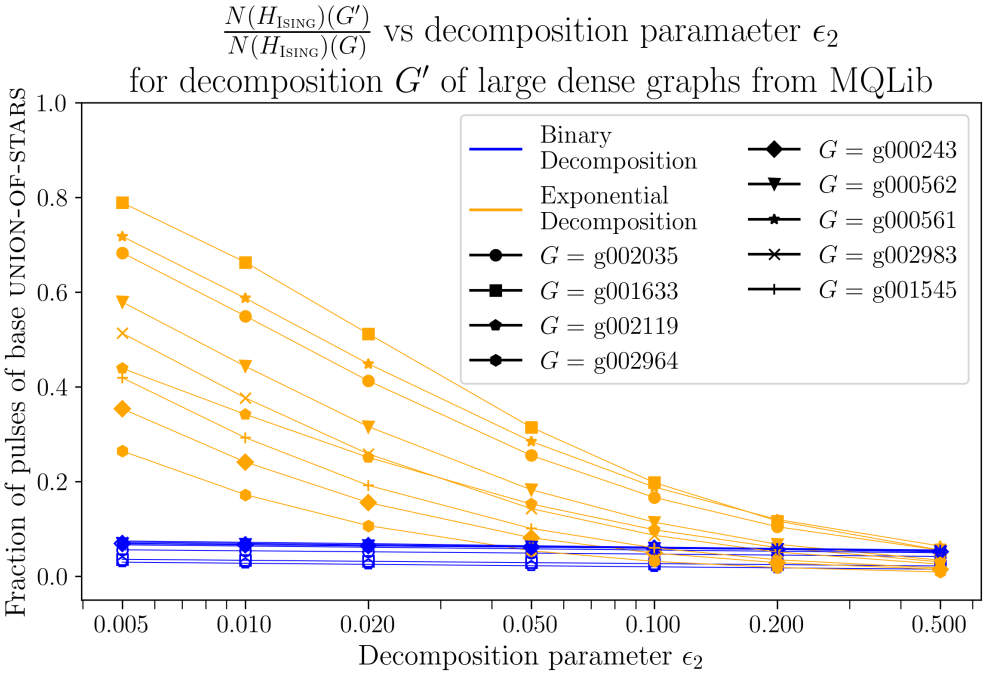


Figure 13: A comparison of $N(H_{\text{ISING}})(G')/N(H_{\text{ISING}})(G)$ (lower is smaller) for decomposed graph G' produced using BINARY-DECOMPOSE and EXP-DECOMPOSE for nine large and dense graphs in MQLib. Note that as $\epsilon_2 \to 0$, BINARY-DECOMPOSE starts to outperform EXP-DECOMPOSE significantly. Additionally, for all graphs BINARY-DECOMPOSE takes fewer than 0.1 of the H_{Ising} pulses taken by base UNION-OF-STARS even for $\epsilon_2 = 0.005$, i.e., with Max-Cut approximation guaranteed to be preserved by more than 99.5%.

these into the otherwise continuous evolution under H_{Ising} to obtain a UNION-OF-STARS trajectory

$$|\psi(t;T)\rangle = \mathcal{T}\left(e^{-iH_{\text{eff}}t}\prod_i \mathcal{J}_{u_i}\left\{\prod_a\left[\prod_{u\in B_a}\sigma_u^x\right]\right\}\right)|\psi(0)\rangle \tag{18}$$

where B_a is the set of qubits which are bit-flipped in the ath step of UNION-OF-STARS. The key point to simplifying this trajectory is to realize that $\sigma_u^z\sigma_u^x = -\sigma_u^x\sigma_u^z$, so it is equivalent to commute all the jump operators $\mathcal{J}_{u_i}$ to the right to obtain

$$|\psi(t;T)\rangle = \mathcal{T}\left(e^{-iH_{\text{eff}}t}\left\{\prod_a\left[\prod_{u\in B_a}\sigma_u^x\right]\right\}\right)\left(\pm\prod_i \mathcal{J}_{u_i}|\psi(0)\rangle\right) \tag{19}$$

The unitary operator on the left is simply the desired Hamiltonian evolution from UNION-OF-STARSalong with a nonunitary factor $\exp(-\Gamma t/8)$ related to the definition of the effective Hamiltonian,

$$\mathcal{T}\left(e^{-iH_{\text{eff}}t}\left\{\prod_a\left[\prod_{u\in B_a}\sigma_u^x\right]\right\}\right) = e^{-i\gamma(t)C'}e^{-\Gamma t/8}, \tag{20}$$

where $\gamma(t) = t/t_{\gamma=1}$ with $t_{\gamma=1}$ the amount of time it takes to compile $\exp(-iC')$ (unitaries $\exp(-i\gamma C')$ are prepared by linearly scaling each step in the compilation by $t/t_{\gamma=1}$), while the jump operators define a trajectory-dependent effective initial state

$$|\psi(0;T)\rangle = \prod_i \mathcal{J}_{u_i}|\psi(0)\rangle \tag{21}$$

where we have set the physically-irrelevant global phase factor $\pm$ equal to unity. To abbreviate notation we drop the time argument in $\gamma(t)$ below, until the final results are presented.

Following the approach of Ref. [20], we will compute spin-spin correlation functions along a single trajectory, then average over the ensemble of trajectories to obtain exact results from the master equation (13). We consider spin-spin correlation functions in terms of raising and lowering operators, $\sigma^+ = |0\rangle\langle 1|$ and $\sigma^- = |1\rangle\langle 0|$ respectively (which are components of σ^y and σ^x), as well as σ^z. We will relate these to QAOA expectation values in the next subsection.

Begin with the correlation function

$$\langle \sigma_u^+ \sigma_v^+ \rangle_T = \frac{\langle \psi(t;T)|\sigma_u^+\sigma_v^+|\psi(t;T)\rangle}{\langle \psi(t;T)|\psi(t;T)\rangle} = \langle \psi_o(0;T)|e^{i\gamma C'}\sigma_u^+\sigma_v^+ e^{-i\gamma C'}|\psi_o(0;T)\rangle \tag{22}$$

where on the far right we have defined $|\psi_o(0;T)\rangle = \prod_i \sigma_{u_i}^z |\psi(0)\rangle$ as a normalized version of $|\psi(0;T)\rangle$; the nonnormalized factors $\sqrt{\Gamma/8}$ from the jump operators $\mathcal{J}$ will be included again later, in the probability definitions $\Pr(\mathcal{F})$ below, following the approach of Ref. [20]; see also the alternative derivation in Ref. [34]. eqn. (22) can be simplified by considering the middle term

$$e^{i\gamma C'}\sigma_u^+\sigma_v^+ e^{-i\gamma C'} = |0_u, 0_v\rangle\langle 1_u, 1_v| e^{i2\gamma \sum_{\mu\neq u,v}(c'_{\mu u}+c'_{\mu v})\sigma_\mu^z} \tag{23}$$

Then we have

$$\langle \sigma_u^+ \sigma_v^+ \rangle_T = \langle \psi_o(0;T)| \left(|0_u, 0_v\rangle\langle 1_u, 1_v| e^{i2\gamma \sum_{\mu\neq u,v}(c'_{\mu u}+c'_{\mu v})\sigma_\mu^z} \right) |\psi_o(0;T)\rangle \tag{24}$$

Finally, noting that $|\psi(0)\rangle = |+\rangle^{\otimes n}$ we have

$$\langle \sigma_u^+ \sigma_v^+ \rangle_T = \frac{(-1)^{\mathcal{F}_u+\mathcal{F}_v}}{4} \prod_{\mu\neq u,v} \cos(2\gamma(c'_{\mu u} + c'_{\mu v})) \tag{25}$$

where $\mathcal{F}_u(T)$ and $\mathcal{F}_v(T)$ are the numbers of jump operators applied to u and v in the trajectory T.

Similarly we have

$$\langle \sigma_u^+ \sigma_v^- \rangle_T = \frac{(-1)^{\mathcal{F}_u(T)+\mathcal{F}_v(T)}}{4} \prod_{\mu\neq u,v} \cos(2\gamma(c'_{\mu u} - c'_{\mu v})) \tag{26}$$

$$\langle \sigma_u^- \sigma_v^+ \rangle_T = \frac{(-1)^{\mathcal{F}_u(T)+\mathcal{F}_v(T)}}{4} \prod_{\mu\neq u,v} \cos(2\gamma(-c'_{\mu u} + c'_{\mu v})) \tag{27}$$

$$\langle \sigma_u^- \sigma_v^- \rangle_T = \frac{(-1)^{\mathcal{F}_u(T)+\mathcal{F}_v(T)}}{4} \prod_{\mu\neq u,v} \cos(2\gamma(-c'_{\mu u} - c'_{\mu v})) \tag{28}$$

Also

$$\begin{aligned}
\langle \sigma_u^z \sigma_v^+ \rangle_T &= \langle \psi_o(0;T)|e^{i\gamma C'}\sigma_u^z\sigma_v^+ e^{-i\gamma C'}|\psi_o(0;T)\rangle \\
&= \frac{\langle 0_u| + \langle 1_u|}{\sqrt{2}} \sigma_u^z e^{i2\gamma c_{uv}\sigma_u^z} \frac{|0_u\rangle + |1_u\rangle}{\sqrt{2}} \frac{(-1)^{\mathcal{F}_v(T)}}{2} \prod_{\mu\neq u,v} \frac{e^{i2\gamma c'_{uv}} + e^{-i2\gamma c'_{uv}}}{2} \\
&= i\sin(2\gamma c'_{uv}) \frac{(-1)^{\mathcal{F}_v(T)}}{2} \prod_{\mu\neq u,v} \cos(2\gamma c'_{\mu,v}),
\end{aligned} \tag{29}$$

$$\langle \sigma_u^z \sigma_v^- \rangle_T = -i\sin(2\gamma c'_{uv}) \frac{(-1)^{\mathcal{F}_v(T)}}{2} \prod_{\mu\neq u,v} \cos(2\gamma c'_{\mu,v}), \tag{30}$$

noting that there is phase factor missing in eqn. (A8) of [20] which is corrected above. Finally

$$\langle \sigma_u^z \sigma_v^z \rangle_T = \langle \psi(0) | \sigma_u^z \sigma_v^z | \psi(0) \rangle = 0. \tag{31}$$

The final step is to average the correlation functions over all possible trajectories T. Following [20], the dephasing terms for each qubit u added randomly following a Poisson process with probability distribution $\Pr(\mathcal{F}_u) = e^{-\Gamma t/4}(\Gamma t/4)_u^{\mathcal{F}}/\mathcal{F}_u!$. The exact spin-spin correlation functions come from averaging over this probability distribution, for example

$$\langle \sigma_u^+ \sigma_v^+ \rangle = \sum_T \Pr(T) \langle \sigma_u^+ \sigma_v^+ \rangle_T = \sum_{\mathcal{F}_u, \mathcal{F}_v = 0} \Pr(\mathcal{F}_u) \Pr(\mathcal{F}_v) \frac{(-1)^{\mathcal{F}_u + \mathcal{F}_v}}{4} \prod_{\mu \neq u,v} \cos(2\gamma(c'_{\mu u} + c'_{\mu v})) \tag{32}$$

and using the identity

$$\sum_{\mathcal{F}=0}^{\infty} \Pr(\mathcal{F})(-1)^{\mathcal{F}} = e^{-\Gamma t/2} \tag{33}$$

we arrive at

$$\begin{aligned}
\langle \sigma_u^+ \sigma_v^+ \rangle = \langle \sigma_u^- \sigma_v^- \rangle &= \frac{e^{-\Gamma t}}{4} \prod_{\mu \neq u,v} \cos(2\gamma(c'_{\mu u} + c'_{\mu v})) \\
\langle \sigma_u^+ \sigma_v^- \rangle = \langle \sigma_u^- \sigma_v^+ \rangle &= \frac{e^{-\Gamma t}}{4} \prod_{\mu \neq u,v} \cos(2\gamma(c'_{\mu u} - c'_{\mu v})) \\
\langle \sigma_u^z \sigma_v^{\pm} \rangle &= \pm i \sin(2\gamma c'_{uv}) \frac{e^{-\Gamma t/2}}{2} \prod_{\mu \neq u,v} \cos(2\gamma c'_{\mu,v})
\end{aligned} \tag{34}$$

C.1 Translation into QAOA expectation values

For QAOA we want to compute expectation values

$$\langle \psi(0) | e^{i\gamma C'} e^{i\beta B} \sigma_u^z \sigma_v^z e^{-i\beta B} e^{-i\gamma C'} | \psi(0) \rangle \tag{35}$$

where $B = \sum_u \sigma_u^x$. The QAOA and Ising expectation values are related as

$$\langle \sigma_u^z \sigma_v^z \rangle_{\text{QAOA}} = \langle e^{i\beta B} \sigma_u^z \sigma_v^z e^{-i\beta B} \rangle \tag{36}$$

where the expectation on the right is with respect to the Ising evolution described earlier. In terms of the Ising evolution we need to compute the expectation value of the operator

$$\begin{aligned}
e^{i\beta B} \sigma_u^z \sigma_v^z e^{-i\beta B} &= e^{i\beta \sigma_u^x} \sigma_u^z e^{-i\beta \sigma_u^x} e^{i\beta \sigma_v^x} \sigma_v^z e^{-i\beta \sigma_v^x} \\
&= (\cos(2\beta)\sigma_u^z + \sin(2\beta)\sigma_u^y)(\cos(2\beta)\sigma_v^z + \sin(2\beta)\sigma_v^y) \\
&= \cos^2(2\beta)\sigma_u^z \sigma_v^z + \cos(2\beta)\sin(2\beta)(\sigma_u^y \sigma_v^z + \sigma_u^z \sigma_v^y) + \sin^2(2\beta)\sigma_u^y \sigma_v^y
\end{aligned} \tag{37}$$

Noting $\langle \sigma_u^z \sigma_v^z \rangle = 0$, taking $\cos(2\beta)\sin(2\beta) = \sin(4\beta)/2$, and replacing $\sigma_u^y = -i\sigma_u^+ + i\sigma_u^-$ we have

$$\begin{aligned}
\langle e^{i\beta B} \sigma_u^z \sigma_v^z e^{-i\beta B} \rangle = &- i\frac{\sin(4\beta)}{2}(\langle \sigma_u^+ \sigma_v^z \rangle - \langle \sigma_u^- \sigma_v^z \rangle + \langle \sigma_u^z \sigma_v^+ \rangle - \langle \sigma_v^z \sigma_u^- \rangle) \\
&+ \sin^2(2\beta)(\langle \sigma_u^+ \sigma_v^- \rangle + \langle \sigma_u^- \sigma_v^+ \rangle - \langle \sigma_u^+ \sigma_v^+ \rangle - \langle \sigma_u^- \sigma_v^- \rangle)
\end{aligned} \tag{38}$$

From above

$$-i(\langle \sigma_u^+ \sigma_v^z \rangle - \langle \sigma_u^- \sigma_v^z \rangle) = -i \left[i \sin(2\gamma c'_{uv}) \frac{e^{-\Gamma t/2}}{2} \prod_{\mu \neq u,v} \cos(2\gamma c'_{\mu v}) + i \frac{e^{-\Gamma t/2}}{2} \sin(2\gamma c'_{uv}) \prod_{\mu \neq u,v} \cos(2\gamma c'_{\mu v}) \right]$$

$$= e^{-\Gamma t/2} \sin(2\gamma c'_{uv}) \prod_{\mu \neq u,v} \cos(2\gamma c'_{\mu v}) \tag{39}$$

$$-i(\langle \sigma_u^z \sigma_v^+ \rangle - \langle \sigma_u^z \sigma_v^- \rangle) = e^{-\Gamma t/2} \sin(2\gamma c'_{uv}) \prod_{\mu \neq u,v} \cos(2\gamma c'_{\mu u}) \tag{40}$$

$$\langle \sigma_u^+ \sigma_v^- \rangle + \langle \sigma_u^- \sigma_v^+ \rangle = \frac{e^{-\Gamma t}}{2} \prod_{\mu \neq u,v} \cos(2\gamma (c'_{\mu u} - c'_{\mu v})) \tag{41}$$

$$\langle \sigma_u^+ \sigma_v^+ \rangle + \langle \sigma_u^- \sigma_v^- \rangle = \frac{e^{-\Gamma t}}{2} \prod_{\mu \neq u,v} \cos(2\gamma (c'_{\mu u} + c'_{\mu v})) \tag{42}$$

Then we have

$$\begin{aligned} \langle e^{i\beta B} \sigma_u^z \sigma_v^z e^{-i\beta B} \rangle = {} & \frac{\sin(4\beta) \sin(2\gamma c'_{uv}) e^{-\Gamma t/2}}{2} \left(\prod_{\mu \neq u,v} \cos(2\gamma c'_{\mu v}) + \prod_{\mu \neq u,v} \cos(2\gamma c'_{\mu u}) \right) \\ & - \frac{\sin^2(2\beta) e^{-\Gamma t}}{2} \left(\prod_{\mu \neq u,v} \cos(2\gamma (c'_{\mu u} + c'_{\mu v})) - \prod_{\mu \neq u,v} \cos(2\gamma (c'_{\mu u} - c'_{\mu v})) \right) \end{aligned} \tag{43}$$

For a QAOA cost Hamiltonian $C = \sum_{u<v} c_{uv} \sigma_u^z \sigma_v^z$ (which may differ from the C' depending on the compilation approach) the total cost expectation is

$$\begin{aligned} \langle C \rangle_{\mathrm{QAOA}} = {} & \sum_{u<v} \frac{c_{uv} \sin(4\beta) \sin(2\gamma(t) c'_{uv}) e^{-\Gamma t/2}}{2} \left(\prod_{\mu \neq u,v} \cos(2\gamma(t) c'_{\mu v}) + \prod_{\mu \neq u,v} \cos(2\gamma(t) c'_{\mu u}) \right) \\ & - \sum_{u<v} \frac{c_{uv} \sin^2(2\beta) e^{-\Gamma t}}{2} \left(\prod_{\mu \neq u,v} \cos(2\gamma(t)(c'_{\mu u} + c'_{\mu v})) - \prod_{\mu \neq u,v} \cos(2\gamma(t)(c'_{\mu u} - c'_{\mu v})) \right) \end{aligned} \tag{44}$$

where we have explicitly noted the time argument in $\gamma(t) = t/t_{\gamma=1}$. Terms related to $\sigma^z\sigma^y$ decay at single-particle rates $\Gamma t/2$, while those related to $\sigma^y\sigma^y$ decay at twice that rate. In the limit $\Gamma \to 0$ and when $C = C'$, the expression (44) agrees with the generic QAOA expectation value in eqn. (14) of Ref. [44].