



Line-Graph Qubit Routing

JORIS KATTEMÖLLE, Department of Physics, University of Konstanz, Konstanz, Germany

SEENIVASAN HARIHARAN, University of Amsterdam, Amsterdam, Netherlands and Centrum Wiskunde en Informatica, Amsterdam, Netherlands

One limitation of current quantum hardware is the restricted connectivity between qubits, as described by the hardware's coupling graph. To overcome this limitation, efficient qubit routing strategies are necessary. We introduce line-graph qubit routing, which routes circuits defined on line graphs to hardware with a heavy coupling graph. We implement line-graph qubit routing and demonstrate its effectiveness in mapping quantum circuits defined kagome, checkerboard, and shuriken lattices to hardware with heavy-hex, heavy-square, and heavy-square-octagon coupling graphs, respectively. Benchmarking shows the ability of line-graph qubit routing to outperform established general-purpose methods in a fraction of the computational time, while offering a depth reduction by up to a factor of 5. Line-graph qubit routing has direct applications in the quantum simulation of lattice-based models, serves as a suitable benchmark for other routing methods, and aids the exploration of the capabilities of near-term quantum hardware.

CCS Concepts: • **Hardware** → **Quantum computation**; **Emerging languages and compilers**; • **Computer systems organization** → **Quantum computing**; • **Computing methodologies** → **Quantum mechanic simulation**;

Additional Key Words and Phrases: Qubit routing, qubit mapping, layout synthesis, line-graphs, quantum simulation, transpilation

ACM Reference Format:

Joris Kattemölle and Seenivasan Hariharan. 2025. Line-Graph Qubit Routing. *ACM Trans. Quantum Comput.* 6, 3, Article 22 (June 2025), 18 pages. <https://doi.org/10.1145/3733842>

1 Introduction

Quantum computing offers potential to revolutionize a wide range of domains by efficiently solving problems that are intractable for classical computers [9, 36]. To run any quantum algorithm, it must be compiled into a quantum circuit that can be executed on the quantum hardware. The hardware coupling graph of a quantum computer, which defines adjacency between qubits based on the ability to perform two-qubit gates between them, plays a crucial role in this process. The problem of ensuring that a quantum circuit is compatible with the hardware coupling graph is referred to as the qubit routing problem [12, 14]. While generalized methods exist for qubit routing [1], a standard approach to implement two-qubit gates between non-adjacent qubits is to insert

J.K. acknowledges funding from the Competence Center Quantum Computing Baden-Württemberg, under the project QORA. Benchmarking was carried out on the Scientific Compute Cluster of the University of Konstanz (SCCKN).

Authors' Contact Information: Joris Kattemölle, Department of Physics, University of Konstanz, Konstanz, Germany; e-mail: physics@kattemolle.com; Seenivasan Hariharan, University of Amsterdam, Amsterdam, Noord-Holland, Netherlands and Centrum Wiskunde en Informatica, Amsterdam, Noord-Holland, Netherlands; e-mail: s.hariharan@uva.nl.



This work is licensed under a Creative Commons Attribution 4.0 International License.

© 2025 Copyright held by the owner/author(s).

ACM 2643-6817/2025/06-ART22

<https://doi.org/10.1145/3733842>

SWAP gates, making the qubits effectively adjacent [12, 14, 27, 33, 54, 58]. To obtain a practical quantum advantage on **noisy intermediate-scale quantum (NISQ)** [45] devices, it is imperative that overhead arising from compilation is kept to a minimum [17, 32]. However, finding the swapping strategy that requires the least number of SWAP gates is NP-hard [5, 12, 21, 33, 56]. Heuristic, probabilistic methods have been developed, but their classical runtime may become problematic for large circuits, and the solution they find may be far from optimal [14, 27, 54, 58]. Striking the right balance between the classical resources required for routing and minimizing the circuit depth of the routed quantum circuit is crucial in maximizing the performance of NISQ devices.

The qubit routing problem is particularly evident in the quantum simulation of lattice-based spin models. One of the first areas in which it was realized that quantum computers could outperform classical computers was that of quantum simulation [15]. When applied specifically to lattice-based spin systems with two-body interactions, quantum simulation by Trotterization [31] approximates the overall time evolution operator of the quantum-mechanical system by a sequence of two-qubit gates, where each two-qubit gate corresponds to the time evolution according to one two-body term in the Hamiltonian (dynamic quantum simulation) [8, 24]. Additionally, by introducing variable parameters for the per-term evolution times, these circuits are transformed to circuits that prepare ansatz states for the **variational quantum eigensolver (VQE)** [57], designed to variationally find the ground state of the quantum-mechanical system (static quantum simulation). Before any routing, these circuits for dynamic and static quantum simulation naturally require hardware with a coupling graph that is equal to the lattice graph of the lattice-based spin system [57]. However, there will generally be a mismatch between the lattice graph of the spin system and the hardware coupling graph. Efficient qubit routing plays a crucial role in overcoming this mismatch.


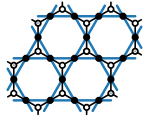
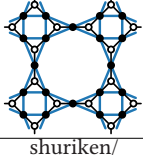
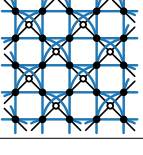
In this article, we develop a qubit routing strategy, which we call line-graph qubit routing, or line-graph routing for short. Let $G = (V(G), E(G))$ be a simple graph. Line-graph routing maps any circuit on a line graph $L(G)$, the *virtual graph*, to hardware with *coupling graph* $\text{heavy}(G)$. Here, $\text{heavy}(G)$ is obtained from the graph G by placing a node on every edge of G . We call these added nodes the heavy nodes of $\text{heavy}(G)$. By definition, the nodes of the line graph $L(G)$ consist of the heavy nodes of $\text{heavy}(G)$. In $L(G)$, two nodes are adjacent if the associated edges in G are incident on the same node of G . Line-graph routing is well-defined for infinite graphs G , although in practice G will be finite.

As a rudimentary example of a graph, its heavy graph and its line graph, take the graph G to be a path graph with three nodes, e.g., the graph with nodes $V(G) = \{1, 3, 5\}$ and the edges $E(G) = \{\{1, 3\}, \{3, 5\}\}$. Then, the graph $\text{heavy}(G)$ has nodes $V(\text{heavy}(G)) = \{1, \dots, 5\}$ and edges $\{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}\}$. The graph $L(G)$ has nodes $V(L(G)) = \{2, 4\}$ and edges $E(L(G)) = \{\{2, 4\}\}$. Some nontrivial examples of line graphs and heavy graphs are given in Table 1. For more information on line graphs, we refer the reader to References [3, 19].

Line-graph routing is efficient, deterministic and provides high-quality solutions. For these reasons, its applicability goes beyond the routing of quantum circuits in practical use cases. First, it provides a suitable benchmark solution against which general-purpose routing algorithms can be tested. Second, given hardware with a fixed coupling graph, line-graph routing naturally defines a class of circuits that be run on that hardware with low overhead. Thereby, it provides means of exploring the capability of noisy, near-term quantum hardware beyond circuits that can be implemented on that hardware natively.

The remainder of this article is organized as follows. We first introduce line-graph routing by example, mapping circuits for the quantum simulation of the **kagome heisenberg antiferromagnet (kagome HAFM)** to hardware with a heavy-hex coupling graph (Section 2). We formalize and generalize this approach to arbitrary circuits and arbitrary line graphs in Section 3. We benchmark

Table 1. Examples of Line-Graph Routing

				
$L(G)$	complete	kagome	shuriken/ square-kagome	checkerboard
heavy(G)	(heavy)-star	heavy-hex	heavy-square-octagon	heavy-square/ Lieb lattice [29]
(model) material	spin glasses [43, 51]	herbertsmithite [42]	atlasovite-like [18, 52]	planar pyrochlore [4, 16, 53]

Line-graph routing maps any circuit with coupling graph $L(G)$ (blue edges) to circuits with coupling graph heavy(G) (black and light gray edges). The light gray edges and nodes need not be available on the hardware the circuit is mapped to. Line-graph routing finds direct application in the quantum simulation of the magnetic properties of some (model) materials (last row) with coupling graph $L(G)$ on hardware with coupling graph heavy(G).

our software implementation of the general algorithm against existing qubit routing approaches in Section 4, to conclude with a discussion and outlook in Section 5.

2 Kagome to Heavy-Hex

Line-graph routing is arguably most clearly explained with an example, which we do in this section by mapping circuits for the quantum simulation of the kagome HAFM to quantum hardware with a heavy-hex coupling graph. First, we use this example due to the relevance of the kagome spin model in exploring quantum phenomena like topological states of matter and quantum spin liquids [42, 49]. The kagome lattice's significance extends to chemistry, as it is frequently observed in transition metal compounds and metal organic frameworks [10, 41]. The ground state of the kagome HAFM is a long-standing open problem in quantum magnetism [26] that can potentially be solved on NISQ devices [24]. By classical emulation of noiseless quantum computers, it was previously demonstrated that the ground-state energy found by a VQE approaches the true ground-state energy exponentially as a function of the circuit depth [24].

Second, the heavy-hex coupling graph is the coupling graph of IBM's current and future superconducting hardware [6, 39]. Among the emerging quantum hardware platforms, IBM's superconducting qubits have gained significant attention due to their rapid development and scalability in the NISQ era. To optimize this superconducting qubit hardware and mitigate the occurrence of frequency collisions and crosstalk [7, 20, 48], error correcting codes are designed on low-degree graphs such as heavy-hex and heavy-square lattices, preventing errors during program execution [6, 11, 39]. This motivates the further development of hardware with these types of coupling graphs.

The relevance of the kagome-to-heavy-hex mapping was further highlighted by the IBM Quantum's Open Science Prize 2022, where the challenge was to prepare the ground state of the kagome HAFM using a VQE and implement it on a 16-qubit IBM Quantum Falcon device with a heavy-hex coupling graph [25]. It is important to note that, also within the context of quantum simulation, the routing problem is not unique to the quantum simulation of spin problems on the kagome lattice. Other lattice-based spin models, such the HAFM on the shuriken lattice, are also known for their geometric frustration and challenging simulation [59].

2.1 Application: Quantum Simulation

One immediate application of the kagome-to-heavy-hex mapping is in the quantum simulation of the kagome HAFM on heavy-hex quantum hardware. In units where $\hbar = 1$, the HAFM has a

Hamiltonian

$$H = \sum_{(i,j)} H_{ij}, \quad H_{ij} = X_i X_j + Y_i Y_j + Z_i Z_j, \quad (1)$$

where the sum is over all edges (i, j) of a given graph. In the current section, this graph could be any patch of the kagome lattice. Here, X_i denotes the Pauli- X operator acting on qubit i (similarly for Y_i, Z_i). This Hamiltonian is straightforwardly generalized to arbitrary two-spin interactions along the edges of a graph,¹ which would conceptually not change the constructions that follows.

The goal of dynamic quantum simulation is to compute expectation values of observables with respect to the time-evolved state $|\psi(t)\rangle = e^{-iHt} |\psi(0)\rangle$, given some initial state $|\psi(0)\rangle$. On a quantum computer, this can be achieved by applying the unitary $\prod_{ij} e^{-iH_{ij}(t/r)}$ to $|\psi(0)\rangle$ (the latter of which is assumed to be easy to prepare) a total of r times. That is,

$$|\psi(t)\rangle \approx \left(\prod_{ij} e^{-i\frac{t}{r} H_{ij}} \right)^r |\psi(0)\rangle. \quad (2)$$

The error in this approximation is of the order t^2/r [31], assuming a perfect quantum computer. After the preparation of $|\psi(t)\rangle$, expectation values can be extracted by repeated preparation and measurements. Note that H_{ij} acts on two qubits, so that $e^{-iH_{ij}(t/r)}$ is a two-qubit unitary that can be decomposed into a few one and two-qubit gates that act on qubits i, j only. In the case of the HAFM, $e^{-iH_{ij}(t/r)}$ is called the HEIS gate [24]

$$\text{HEIS}_{ij}(\alpha) \equiv e^{-i\alpha/4} e^{-i\alpha H_{ij}/4}, \quad (3)$$

where, in anticipation of static simulation, we have set $\alpha = 4t/r$, and where the physically irrelevant prefactor $e^{-i\alpha/4}$ and a factor of $1/4$ in the exponent are included for consistency with Reference [24].

We can go from the circuit for dynamical simulation [Equation (2)] to the circuits needed for static quantum simulation by a VQE [9, 34, 44] by considering α as a free parameter in every occurrence of the HEIS gate. VQEs form a promising method for computing ground state energies of various many-body systems in the NISQ era [9, 34, 44]. In VQEs, a parameterized quantum state is prepared with a parameterized circuit. The energy of this state is measured and optimized using a classical heuristic optimization routine. By the variational principle, the lowest energy that is found in this way provides an upper bound on the ground-state energy. In principle, there are no a priori restrictions on the structure of the ansatz circuit [44]. In the subclass of VQEs using the so-called **Hamiltonian variational ansatz (HVA)**, however, each gate in the parameterized circuit is either formed by parameterized time evolution along a term in the Hamiltonian, or by parameterized time evolution according to some reference Hamiltonian [57]. The initial state of the circuit is the (known and easy-to-prepare) ground state of the reference Hamiltonian.

One possible quantum circuit following from the above considerations is depicted in Figure 1 (colored edges). This circuit can both be used for the dynamic quantum simulation (fixed parameters α) or for static quantum simulation using the HVA (free parameters α , to be optimized by a classical optimization routine). Here, every color depicts a different layer of the circuit. In the first layer, singlet states $(|01\rangle - |10\rangle)/\sqrt{2}$ are placed along the blue edges. This provides the initial state of either the circuit for dynamical or static quantum simulation. It is the ground state of a reference Hamiltonian $\sum_{(i,j)} H_{ij}$, where the sum is over the blue edges (i, j) . After preparation of the initial

¹These are described by $H = \sum_{(i,j)} P_{ij} + \sum_i P_i$, with two-spin terms $P_{ij} = \sum_{kl} Y_{ij}^{(kl)} \sigma_i^{(k)} \sigma_j^{(l)}$ and single-spin terms $P_i = \sum_k Y_i^{(k)} \sigma_i^{(k)}$, where $\{\sigma_i^{(k)}\}_{k=0}^3 = \{\mathbb{1}, X_i, Y_i, Z_i\}$.

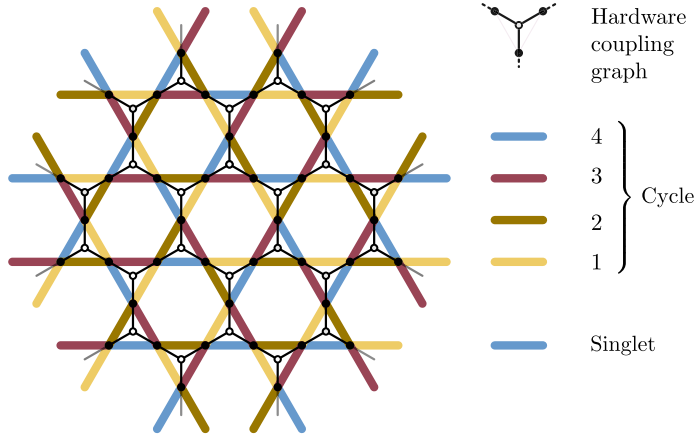


Fig. 1. The kagome lattice (colored edges) is the line graph of the hexagonal lattice (black edges). As shown in the figure, the nodes of the kagome lattice can be identified with the heavy nodes of the heavy-hex lattice. The colored edges represent one possible circuit with a kagome coupling graph. Every color represents a layer in this circuit. In the first layer, singlet states are created along the blue lines. Thereafter, HEIS gates [Equation (3)] are applied along all colored edges in sequence, defining one circuit cycle. This cycle is repeated to obtain the complete circuit. Figure adapted from Reference [24].

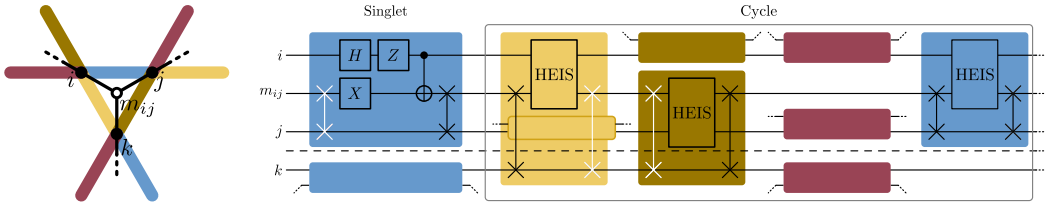


Fig. 2. The line-graph routed quantum circuit for the quantum simulation of the kagome HAFM (Figure 1), focusing on one triangle. The circuits involving the other triangles are similar. Empty colored boxes indicate subcircuits coming in from other triangles. The white SWAP gates can be omitted by a relabeling of the qubits (first SWAP gate) and the cancellation of double SWAP gates [other white SWAP gates, cf. Equation (5)].

state, HEIS gates are placed along all orange, green, red, and blue edges. This combination of HEIS gates defines one *cycle* of the circuit. The cycle is repeated r times. (The color blue defines both the initial state and the last layer of the cycle.) Alternative circuits for dynamic or static quantum simulation (using the HVA) may be achieved by changing the initial state or the order of the HEIS gates.

Line-graph routing maps the kagome circuit from Figure 1 to a circuit on the heavy-hex lattice by first identifying nodes of the respective lattices, as in Figure 1. Subsequently, Equation (4) is applied to all gates, after which superfluous SWAP gates are removed [Equation (5)]. The routed circuit thus obtained is shown in Figure 2.

2.2 Line-Graph Routing

The first step in line-graph routing is establishing a one-to-one correspondence between the nodes of the virtual graph (in this section, the kagome lattice) and the hardware coupling graph (in this section, the heavy-hex lattice). This correspondence is achieved by aligning the nodes of the kagome lattice with the heavy nodes of the heavy-hex lattice, as shown in Figure 1. Subsequently,

the light (non-heavy) nodes of the heavy-hex lattice are used to mediate two-qubit gates between the spins on the nodes of the kagome lattice.

To see this in more detail, assume a kagome quantum circuit, that is, a circuit composed of single-qubit gates on qubits i and two-qubit gates along the edges (i, j) of a patch of the kagome lattice. To map the circuit from the kagome to the heavy-hex lattice, we label the heavy qubits on the heavy-hex lattice with the labels i of the congruent qubits on the kagome lattice, as shown in Figure 1. Under this identification, any single-qubit gate in the kagome circuit is trivially mapped to a single-qubit gate on the heavy-hex lattice.

To map the two-qubit gates, let us label the ℓ th two-qubit gate in the kagome circuit, acting on qubits (i, j) , by U_{ij}^ℓ . Any such gate can be performed on the heavy-hex lattice by mapping it to the three-qubit *mediated two-qubit gate* MU

$$U_{ij}^\ell \mapsto MU_{imj}^\ell = \text{SWAP}_{mi} U_{mj}^\ell \text{SWAP}_{im}, \quad (4)$$

where qubit $m = m_{ij}$ mediates the interaction between qubits i and j . This map provides the cornerstone of line-graph routing. The key point of line-graph routing is that for every pair of qubits (i, j) the existence and uniqueness of the mediating qubit $m = m_{ij}$ is guaranteed by the definition of $L(G)$ and $\text{heavy}(G)$ (see Section 1).

Equation (4) introduces many SWAP gates that need not be performed physically. First, SWAP gates occurring at the beginning and end of the routed circuit can be accounted for by a relabeling of the qubits. Second, any two consecutive SWAP gates can be cancelled. These double SWAP gates are introduced by Equation (4) if there are two consecutive two-qubit gates acting on the same two qubits (possibly with additional single-qubit gates on those qubits in between). Double SWAP gates are also introduced by Equation (4) in the case of two consecutive two-qubit gates that have a single qubit in common and where the two resulting mediated gates have a mediating qubit in common. That is, if $m = m_{ij} = m_{ik}$, we have by Equation (4) that

$$\begin{aligned} U_{ik}^{\ell'} U_{ij}^\ell &\mapsto (\text{SWAP}_{mi} U_{mk}^{\ell'} \text{SWAP}_{im})(\text{SWAP}_{mi} U_{mj}^\ell \text{SWAP}_{im}) \\ &= \text{SWAP}_{mi} U_{mk}^{\ell'} U_{mj}^\ell \text{SWAP}_{im}. \end{aligned} \quad (5)$$

In summary, line-graph routing first associates the nodes of $L(G)$ with the heavy nodes of $\text{heavy}(G)$, applies the map of Equation (4) to all two-qubit gates, and finally removes superfluous SWAP gates as described above.

3 General Case

Here, we formalize and generalize the routing strategy from the previous section, which leads to our main results, Theorem 3.1 and Algorithm 2. To introduce notation, we now define line- and heavy graphs more formally. The graph $\text{heavy}(G)$ is obtained from G by defining a node i and edges (m, i) , (i, m') for each edge (m, m') in G . The nodes i are referred to as the heavy nodes.

A minor and subtle point is that this construction may lead to fully *inactive mediating nodes*. Consider an induced subgraph P of $\text{heavy}(G)$ with edges of the form $E(P) = \{(m, i), (i, m')\}$, where m and m' are mediating qubits and i a program qubit. For example, in the star graph (Table 1), all paths emanating from the center node (i.e., the “rays” of the star) are of the form of P . Line-graph routing [previewed in Equation (4)] will never use mediating qubits associated with nodes locally equivalent (i.e., including neighbors and next-nearest neighbors) to node m in P and can therefore be discarded without affecting the routed circuit. We call such nodes the inactive mediating nodes. In the example of the star graph (Table 1), this means the qubits associated with the gray nodes may be removed from the routed circuit. To avoid frequent mention of this minor but subtle point, in this article we also refer to heavy graphs where the nodes locally equivalent (including neighbors

and next-nearest neighbors) to node m in P are removed as heavy graphs. After these removals, we still refer to nodes locally equivalent (only including neighbors) to node i in P as heavy nodes.

The line graph of G , $L(G)$, is defined in terms of $\text{heavy}(G)$ as follows: construct $\text{heavy}(G)$ from G and let the node set of $L(G)$ be equal to the set of heavy nodes i of $\text{heavy}(G)$. An edge (i, j) is added to $L(G)$ if and only if there exists a node m in G such that (i, m) and (m, j) are edges in $\text{heavy}(G)$.

We say that a quantum circuit C , consisting of one- and two-qubit gates by assumption, has coupling graph $G = (V, E)$ if V is equal to the set of qubit labels in C and if $(i, j) \in E$ if and only if there exists a two-qubit gate U_{ij} in C . If a *quantum computer* has coupling graph $G = (V, E)$, then it can perform arbitrary single-qubit gates on the qubits associated with each node in V and arbitrary two-qubit gates along each edge in E . By Equation (4), we then have the following theorem.

THEOREM 3.1 (LINE-GRAPH ROUTING). *Every quantum circuit C with coupling graph $L(G)$ can be performed on quantum hardware with coupling graph $\text{heavy}(G)$ with a SWAP overhead of at most two times the number of two-qubit gates in C .*

PROOF. By definition, there is a one-to-one correspondence between the nodes of $L(G)$ and the heavy nodes i of $\text{heavy}(G)$. Therefore, the single-qubit gates of C can be mapped directly to hardware with coupling graph $\text{heavy}(G)$. Furthermore, for every edge (i, j) in $L(G)$, there are edges (i, m) and (m, j) in $\text{heavy}(G)$, where $m = m_{ij}$ can be determined uniquely from i and j . Thus, every two-qubit gate U_{ij} in C can be mapped to the three-qubit gate $MU_{imj} := \text{SWAP}_{mi}U_{mj}\text{SWAP}_{im}$, leading to an overhead of 2λ SWAP gates, with λ the total number of two-qubit gates in C . \square

We note this theorem is theoretically simple. Its strength lies in the fact that it considers the most general type of input circuit in our work (an arbitrary circuit on a line graph) and nevertheless provides a bound of the SWAP-overhead that is constant in any of the properties of $L(G)$, $\text{heavy}(G)$, and G . Moreover, the bound is trivial to compute. Also note that the theorem provides a hierarchy of mappings, as $\text{heavy}(G)$ itself may be a line graph. For example, invoking the above theorem twice gives a routing from $L(L(G))$ to $\text{heavy}(L(G))$.

Theorem 3.1 bounds the number of SWAP gates in the routed circuit, but not the depth of the routed circuit. To obtain a bound on the depth, we consider specifically the quantum simulation task of Section 2.1. Formally, in quantum simulation by trotterization [Equation (2)], the circuit cycle $\prod_{ij} e^{-i\frac{t}{r}H_{ij}}$ contains exactly one two-qubit gate per edge (i, j) on the graph on which H is defined, which, in the context of this article, is a line graph $L(G)$. We call circuits with this property a quantum simulation circuit cycle on $L(G)$. We bound the depth by $\chi(G)$, the chromatic number of G (i.e., the minimum number of colors needed to color the nodes of G so that no two nodes of the same color are adjacent in G), and $\Delta(G)$, the maximum degree of G (i.e., the maximum number of edges incident on any node in G).

THEOREM 3.2. *For any line graph $L(G)$, there exists a quantum simulation circuit cycle on $L(G)$ so that the line-graph routed circuit cycle on $\text{heavy}(G)$ has a depth no more than*

$$\frac{3}{2}\chi(G)\Delta(G)(\Delta(G) - 1). \quad (6)$$

PROOF. Consider any node m in G . Its neighbors \mathcal{N}_m in $\text{heavy}(G)$ are all program qubits i and the induced subgraph of \mathcal{N}_m in $L(G)$ is the complete graph on \mathcal{N}_m . Since this graph has no more than $n := \frac{1}{2}\Delta(G)(\Delta(G) - 1)$ edges, a gate can be placed along each of these edges by a circuit with a depth no more than n . This circuit is routed to (a subgraph of) the graph $\text{heavy}(G)$ by line graph routing. By Equation (4), this routed circuit has a depth no more than $3n$.

Now, consider a minimal node coloring of G with colors $0, \dots, \chi(G) - 1$. For all $m \in V(G)$ with color 0, use the above construction to create a routed circuit effectively implementing gates along

all edges in the induced subgraph of \mathcal{N}_m in $L(G)$, with \mathcal{N}_m the neighbors of m in $\text{heavy}(G)$. All these routed circuits can be performed in parallel since, by the minimal vertex coloring of G , no node in \mathcal{N}_m is also in $\mathcal{N}_{m'}$ for $m, m' \in V(G)$, $m \neq m'$, and m, m' both having color 0.

Repeating the above process for all $\chi(G)$ vertex colors creates the line-graph routed circuit on $\text{heavy}(G)$ that effectively implements gates along all edges of $L(G)$ in a depth no more than $3n\chi(G) = \frac{3}{2}\chi(G)\Delta(G)(\Delta(G) - 1)$. \square

Naturally, p quantum simulation circuit cycles on $L(G)$ can thus be performed in depth $\frac{3}{2}p\chi(G)\Delta(G)(\Delta(G) - 1)$ on $\text{heavy}(G)$. Assume the quantum simulation circuit can be prepared with line graph routing in depth D_{init} on $\text{heavy}(G)$. Furthermore, note that we can remove the initial and final layer of SWAPs by a relabeling of the qubits, giving a depth reduction of 2. Then, a bound for the full quantum simulation task, including state preparation and p circuit cycles, is

$$D_{\text{bound}} = \frac{3}{2}p\chi(G)\Delta(G)(\Delta(G) - 1) + D_{\text{init}} - 2. \quad (7)$$

In practice, one may be given hardware with a coupling graph H' and asked to find the class of circuits that can be run on this hardware using the line-graph construction. (For an overview of the graphs that follow, please see Figure 3.) This task may arise if one has specific but limited quantum hardware available and wants to explore its capabilities by looking at quantum circuits that can be routed to this hardware with low overhead. The most immediate class of such quantum circuits is obviously formed by circuits with coupling graph H' . In case H' is a heavy graph, $H' = \text{heavy}(G)$, line-graph routing extends the possibilities to the class of circuits with coupling graph $L(G)$. If H' is a heavy graph, it is trivial to find G such that $H' = \text{heavy}(G)$. It is also trivial to construct the line graph $L(G)$ from G . Therefore, given a heavy hardware coupling graph, it is trivial to find the class of circuits that can be run on it by line-graph routing. For example, given hardware with a heavy-hex coupling graph, it is trivial to find that line-graph routing yields the class of kagome circuits.

The task can also be reversed: given a circuit with a coupling graph G' , find the hardware on which it can be run with low overhead (cf. Figure 3). Again, the first place to look would be hardware with coupling graph G' . Line-graph routing extends the possibilities by adding hardware with a coupling graph of $\text{heavy}(G)$ with G such that $G' = L(G)$.

But how to find G from the circuit coupling graph G' ? This is less straightforward, first because it is not possible to find G from G' if G' is not a line graph. In this case, line-graph routing cannot be applied. There are numerous straightforward ways of checking whether a graph is a line graph. For example, Beineke's theorem states that a graph is a line graph if and only if it does not contain an induced subgraph out of a set of nine forbidden subgraphs [2]. One of these forbidden graphs is the claw (Υ). For example, since the heavy-hex and hexagonal lattices consist entirely of claws, they are themselves not line graphs. Second, even if G' is a line graph, it is nontrivial to find the graph G such that $G' = L(G)$. Nevertheless, Roussopoulos' algorithm [47] finds G from G' , or reports that G' is not a line graph, in time $O(\max\{n, |E_{G'}|\})$, with n the number of nodes and $|E_{G'}|$ the number of edges of G' .

We briefly introduce the concepts from Roussopoulos' algorithm that are useful to us later. If G' is a line graph, Roussopoulos' algorithm partitions the edges of G' into complete subgraphs in such a way that no node lies in more than two of the subgraphs (which is possible if and only if G' is a line graph). Then, the nodes of G correspond to the sets in the partition. Additionally, the nodes that lie in only one of the sets in the partition are added as nodes of G as sets of length one. Two nodes in G are adjacent if their corresponding sets have a nonempty overlap. For example, in Figure 1, each triangle of colored edges forms a set in the partition of the kagome lattice because triangles are fully connected subgraphs of the kagome lattice and no node of the kagome lattice

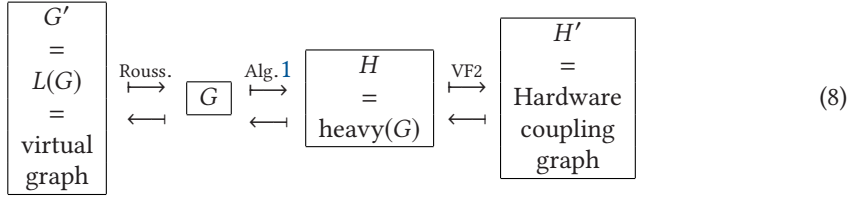


Fig. 3. An overview of the graphs related to line-graph routing. All maps from the right to the left are trivial given that H' is a heavy graph.

is in more than two triangles. Defining the partitions as the nodes of $L(G)$ and putting an edge between two nodes in $L(G)$ whenever the corresponding sets have a nonempty intersection results in the hexagonal lattice (open nodes in Figure 1).

For line-graph routing [as previewed in Equation (4)], $L(G)$ must be mapped to $\text{heavy}(G)$ in such a way that the labels of the heavy nodes of $\text{heavy}(G)$ are identical to the labels of the nodes of $L(G)$. The output of Roussopoulos' algorithm provides a convenient way to achieve this labeling through the following algorithm. It takes a graph G , as generated by Roussopoulos' algorithm, as input and returns $H = \text{heavy}(G)$, a heavy graph where the set of heavy nodes is equal to the set of nodes (i.e., with equal labels) of $G' = L(G)$. This generalizes and automatizes the identification of nodes made in Figure 1.

Algorithm 1 (Congruent Heavy Labels). Consider the edges (a, b) of G for which both a and b contain more than one node. (The nodes in a and b are from G' , the latter of which need not be provided explicitly to the current algorithm.) For all such edges (a, b) , add (a, c) and (c, b) to an empty graph H . Because a and c are distinct sets in a partition of the edges of G' into complete subgraphs and a and c have a nonempty overlap, c contains a single entry and this entry is a node from G' . Now, consider the edge cases, that is, the edges (a, b) in G where either a or b is a set of length one. For every such edge, add (a, b) to H .

Everything is now in place for the succinct presentation of line-graph routing. It maps any circuit C , with unknown coupling graph G' , to a circuit with coupling graph H whenever G' is a line graph, $G' = L(G)$, and where $H = \text{heavy}(G)$.

Algorithm 2 (Line-graph Routing). Construct the coupling graph G' of the circuit C . Run Roussopoulos' algorithm on G' to obtain $G = L^{-1}(G')$. Construct $\text{heavy}(G)$ using Algorithm 1. For each two-qubit gate U_{ij} in C , let $m = m_{ij}$ be the node in $\text{heavy}(G)$ that is in between nodes i, j of $\text{heavy}(G)$ and replace U_{ij} by $MU_{imj} := \text{SWAP}_{mj}U_{im}\text{SWAP}_{jm}$.

Note the existence and uniqueness of mediating qubit m are guaranteed by the definition of line graphs. In some cases, some qubits, related to the so-called lone leaf nodes in the coupling graph of the output circuit of line-graph routing, can be removed from that circuit, reducing the SWAP and qubit count of line-graph routed circuits. This removal leads to a marginal improvement, which may nevertheless be crucial given hardware with few qubits, but arguably obfuscates the general idea of line-graph routing. The definition of lone leaf nodes and the method for their removal is presented in Appendix A. We note these nodes are distinct from the inactive mediating nodes introduced in the beginning of this section. The lone leaf nodes can be removed from the circuit after the removal of the inactive mediating nodes.

As opposed to heuristic methods [27], line-graph routing is deterministic, allowing rigorous performance guarantees. Given the time complexity of Roussopoulos' algorithm [47], it is straightforward to show that the time complexity of line-graph routing is $O(\Lambda^2)$, with Λ the number of

gates in the input circuit C . A tighter bound on the time complexity can possibly be obtained, but this requires a more sophisticated analysis. Such an analysis is unnecessary for the current purposes because a nonoptimized implementation of line-graph routing already routes circuits with thousands of qubits and hundred thousands of gates within a minute on a standard laptop [23].

To utilize line-graph routing in practice, one additional step may be required. Quantum hardware providers use a certain labeling of the qubits on their hardware, leading to a hardware coupling graph H' . This labeling generally differs from the labeling of $H = \text{heavy}(G)$ obtained through Algorithm 1. So, as a final step of line-graph routing, a map is required from the nodes of H to the nodes of H' . Such a relabeling step is typically already performed by quantum hardware providers by default [46], so in the rest of this article, we refer to H as the hardware coupling graph and in this wording leave the possible mapping to the fixed qubit labels provided by a hardware provider implicit.

Let us, nevertheless, discuss the relabeling step for the case of line-graph routing in more detail. Typically, H and H' are both patches of the same lattices depicted in Table 1, so that a relabeling can be found straightforwardly and efficiently by “overlaying” the two patches and identifying overlapping nodes. However, also general-purpose algorithms for this task may be used. For general graphs, mathematically, the relabeling amounts to finding a subgraph isomorphism, which can be performed by algorithms such as VF2 [13]. The subgraph isomorphism problem is NP-complete [13]. However, we find this not to pose a problem in practice [23].

The VF2 algorithm generates a list of isomorphic subgraphs. Thus, some noise-awareness can be achieved in line-graph routing in the last step by choosing the best subgraph out of the isomorphic subgraphs according to a performance metric, such as the average two-qubit gate fidelity on that subgraph.

4 Implementation and Benchmarking

In the Supplemental Material,² we implement, showcase, and benchmark line-graph routing, Algorithm 2, together with the removal of lone leafs (Appendix A). The implementation takes any Qiskit [46] quantum circuit consisting of one- and two-qubit gates, constructs its coupling graph $L(G)$ or reports that the coupling graph is not a line graph, finds G and $\text{heavy}(G)$, and outputs the routed circuit with coupling graph $\text{heavy}(G)$. The implementation does not rely heavily on Qiskit’s methods and can hence straightforwardly be transformed to an implementation in other quantum software development kits.

Line-graph routing is benchmarked against all routing methods available in Qiskit (version 0.42.1) by default [23, 46]. In this section, we show line-graph routing confidently outperforms these default methods on relevant problem instances. There are also problem instances where line-graph routing does not outperform the default methods. In the end of this section, we discuss for which types of instances we expect line-graph routing to be superior. We consider two types of circuits.

(i) *Random*. With probability $2/5$, a CNOT gate is placed along a randomly chosen edge of a given virtual graph. With a probability $3/5$ a gate from the set $\{H, S, T\}$ is chosen uniformly at random and placed at a random node.

(ii) *Quantum simulation*. As described in detail for the kagome lattice (Section 2.1), circuits for the dynamical and static quantum simulation of the HAFM on any lattice can be defined by an edge-coloring of that lattice [8, 24]. An edge coloring of a graph is an assignment of colors to the edges such that no two edges with the same color are incident on the same node. This edge

²Supplemental Material available at <https://github.com/kattemolle/LIGRAR>. In the Supplemental Material, we implement, showcase, and benchmark line-graph routing for various graphs. The Supplemental Material is also available at Reference [23].

Table 2. Excerpt of the Benchmarking Data Available in the Supplemental Material [23]

$L(G)$	Routing method	D_{opt}	n_{SWAP}	n_{qubit}	t_{tot} (s)	\bar{t} (s)	D_{bound}	$n_{\text{SWAP}}^{\text{bound}}$
Complete	line-graph	1036	650	10	<1	<1	N.A.	720
	SABRE	720	146	9	3	<1		
Kagome	line-graph	226	7968	300	43	43	289	10626
	SABRE	790	8486	200	886	55 ± 1		
Shuriken	line-graph	209	13600	476	77	77	289	19404
	SABRE	1099	17064	323	2520	157 ± 4		
Checkerboard	line-graph	435	18521	393	67	67	580	23298
	SABRE	2121	23060	282	1750	109 ± 1		

The column headers are defined in the main text. In case of the complete graph, a random circuit was routed. For the kagome, shuriken and checkerboard graphs, a quantum-simulation circuit was routed. Both types are detailed in the main text.

coloring is called minimal if it uses the least possible number of colors. We perform edge coloring of the virtual lattices by an automatic method that generally finds an edge coloring that is not minimal. The benefit of this coloring method is that it does not require a manual assignment of edge colors. The downside is that we expect line-graph routing to work best (compared to other methods) for circuits derived from a minimal edge coloring. This does not pose a problem because, as we show in this section, line-graph routing is already able to outperform the default methods in Qiskit in routing circuits derived from a nonminimal edge coloring.

We found SABRE [27] to outperform the other methods in Qiskit and therefore we focus on a comparison between line-graph routing and SABRE in what follows. Unlike line-graph routing, SABRE is a probabilistic routing method that obtains a different qubit routing with each run. Additionally, the intensity of the optimization that is part of SABRE can be varied, leading to a tradeoff between the classical resources required and the performance characteristics of the routed circuits. We address these issues by running SABRE 16 times (at fixed optimization level) and comparing the performance against one run of line-graph routing. We do this separately for every optimization level available by default in Qiskit, which range from optimization level 0 (“no optimization”) to optimization level 3 (“heavy-weight optimization”) in integer steps [46].

In Table 2, we show an excerpt of the benchmarking data, which includes problem instances on which line-graph routing does and does not perform well. The following performance characteristics are listed.

(i) D_{opt} . The lowest number of quantum circuit layers in the output of the routing method among all runs. (Line-graph routing is run once per virtual graph, SABRE is run 16 times per virtual graph). Routed circuits are obtained by inserting SWAP gates (as dictated by line-graph routing or SABRE) and no further gate identities are used to simplify the resulting circuits. So, in the case of random input circuits, the routed circuits consist of gates from the set $\{\text{CNOT}, H, S, T\} \cup \{\text{SWAP}\}$. The routed circuits contain, for example, double H gates if those were present in the input circuit. In the case of quantum simulation input circuits, the routed circuits consist of gates from the set $\{\text{SINGLET}, \text{HEIS}(\alpha)\} \cup \{\text{SWAP}\}$.

(ii) n_{SWAP} . The number of SWAP gates of the routed circuit that achieved the lowest depth.

(iii) n_{qubit} . The number of active qubits in the routed circuit that achieved the lowest depth.

(iv) t_{tot} . The total wall-clock time needed to run all runs of the routing method. For line-graph routing, this includes the time needed to find $\text{heavy}(G)$ from $L(G)$. The implementation of line-graph routing repeatedly loops through all gates using (slow) Python loops and can likely be sped up considerably, if needed. We use Qiskit’s standard implementation of SABRE. SABRE is given the target graph $\text{heavy}(G)$ as input and hence finding $\text{heavy}(G)$ from $L(G)$ is not included in its wall-clock time. The benchmarks for different methods are always run on the same hardware.

(v) \bar{t} . The average wall-clock time for a single run of the routing method. The error bars are calculated by bootstrapping the individual wall-clock times and represent symmetrized 95% confidence intervals.

(vi) D_{bound} . The upper bound on the depth of the line-graph routed circuit achieving the same quantum simulation task as given by Equation (7).

(vii) $n_{\text{SWAP}}^{\text{bound}}$. The upper bound on the number of SWAP gates as given by Theorem 3.1

The first two data lines of Table 2 show the performance characteristics of line-graph routing and SABRE when applied to a random circuit containing 900 gates on the complete graph with 9 nodes. SABRE was run with optimization level 1. (Passing a higher optimization level to Qiskit's transpiler will trigger the usage of gate identities.) Already at an optimization level of 1 SABRE outperforms line-graph routing on all performance characteristics considered except the total wall-clock time.

We now consider the ensuing data lines of Table 2. They show benchmarking results when routing circuits for the quantum simulation of the HAFM defined on the kagome, shuriken, and checkerboard lattices. The circuits are 16 circuit cycles deep and cover patches measuring 7×7 unit cells. SABRE was run at Qiskit's transpiler optimization level 3, the highest optimization level available. (No gate identities are used in the transpilation because the gates {SWAP, SINGLET, HEIS} have unknown properties to Qiskit's transpiler [46].) Line-graph routing confidently outperforms SABRE when it comes to the optimal depth, by factors of about 3.5, 5.3, and 4.9, while at the same time using considerably less computational time. Line-graph routing also achieves fewer SWAP gates, but for this performance metric the gain is not so pronounced. On the other hand, line-graph routing uses more qubits than SABRE. Nevertheless, the total space-time volume of the routed circuits remains less for line-graph routing.

Line-graph routing was conceived while keeping in mind its application in mapping quantum circuits to hardware with a lattice-like low-degree coupling graph. It is therefore not expected to perform well in mapping quantum circuits to hardware with a high-degree coupling graph without lattice-like structure. In fact, the benchmarking results show that line-graph routing is not well-suited for mapping circuits on the complete coupling graph to hardware with a star coupling graph. One property of line-graph routing that leads to its low effectivity on unstructured, high-degree graphs is that in the output circuit of line-graph routing, the qubits are assigned a base location where they return to after they are acted on by one [Equation (4)] or multiple [Equation (5)] gates from the input circuit. This is likely advantageous for input quantum circuits with a structured, low-degree coupling graph. For example, in Figure 2, in the third layer of the cycle (red gates), all qubits are still close to where they are needed, leading to the insertion of few SWAP gates to get them there.

However, on high-degree coupling graphs, the property of a base location need not be advantageous since any qubit can be routed to any other qubit in relatively few steps and regularly returning qubits to their base location leads to the insertion of unnecessary SWAP gates. As an extreme example, let us look at the action of line-graph routing on a circuit C with a complete coupling graph, where at one point in C a cascade of CNOT gates, $\prod_{i=n}^1 \text{CNOT}_{i,i+1}$, is prescribed. Line-graph routing maps the circuit C to a circuit on the star graph (Table 1). To perform $\text{CNOT}_{i,i+1}$ on star-graph hardware, line-graph routing first swaps qubit i to the center of the star, performs a CNOT between the center qubit and qubit $i + 1$, and swaps qubit i back to its original position [Equation (4)]. Not insisting that the qubits eventually return to their original position leads to the possibility of more efficient routing strategy. To perform the $\text{CNOT}_{i,i+1}$, swap qubit i to the center, perform the CNOT between the center qubit and qubit $i + 1$, and leave qubit i in the center. After repeating this procedure for the subsequent CNOT in the circuit, qubit i will end up at the initial

location of qubit $i + 1$. The latter approach uses half of the number of SWAP gates compared to line-graph routing.

A second situation in which line-graph routing is not expected to perform well is when there are subgraphs of the virtual graph $L(G)$ that are isomorphic to subgraphs of $\text{heavy}(G)$. In that case, unnecessary mediating qubits may be inserted. For example, let $L(G)$ be the path graph on n nodes. When applied to circuits on this graph, line-graph routing introduces mediating qubits between all qubits of $L(G)$, leading to a circuit on a path graph with $n - 1$ added mediating qubits. This happens despite the fact that no routing is needed at all. If needed, line-graph routing can be enhanced such that it detects and prevents this behavior.

5 Discussion and Outlook

In this article, we developed line-graph routing, a qubit routing strategy that efficiently and deterministically maps any quantum circuit on a line graph $L(G)$ to a circuit on the heavy graph $\text{heavy}(G)$. By software implementation and benchmarking, we showed its ability to outperform standard, general-purpose methods on input quantum circuits, circuit sizes, and hardware coupling graphs of practical relevance.

Line-graph routing showed not to perform well in mapping circuits on the complete graph to the star graph. We attribute this to the high degree and absence of a lattice-like structure of the complete graph. Based on our benchmarking results, we expect line-graph routing to outperform general-purpose methods in routing circuits with a line-graph coupling graph to low-degree hardware coupling graphs. For superconducting qubits, these are exactly the hardware coupling graphs preferable from an engineering standpoint [6, 11, 39].

Line-graph routing is limited in its applicability because it is only defined for pairs of graphs $(L(G), \text{heavy}(G))$. General-purpose methods are able to map any circuit on any graph to a circuit on any other graph (given that the latter graph is connected and has at least the same number of nodes as the former graph). However, in general, finding the optimal routing strategy is NP-hard. It is therefore unlikely that general-purpose methods can find the optimal or close-to-optimal routing strategy. Routing strategies that are defined on a subset of possible input circuits can nevertheless be highly efficient and effective, and indeed line-graph routing is shown to have the ability to outperform standard-purpose methods on the subset of problem instances for which it is defined. The set of pairs of graphs $(L(G), \text{heavy}(G))$ is still infinitely large and contains pairs of practical relevance. Additionally, line-graph routing provides a lower bound for the number of swap gates that need to be inserted. Therefore, it provides a well-defined and efficiently computable benchmark for general-purpose methods.

The routed circuit for the simulation of a nearest-neighbor model on $L(G)$ can be reinterpreted as a circuit for the quantum simulation of a next-nearest-neighbor model on $\text{heavy}(G)$. This is evident from Table 1 and Figure 1 and follows directly from the definition of line graphs as given in Section 3. This routed circuit for the simulation of a nearest-neighbor model on $L(G)$ can be lifted to a circuit for the quantum simulation of a model containing both nearest- and next-nearest-neighbor interactions on $\text{heavy}(G)$ by adding the gates arising from nearest-neighbor interactions on $\text{heavy}(G)$ to the routed circuit. These gates naturally satisfy the $\text{heavy}(G)$ coupling graph and can thus be added to the circuit without any routing. This further enlarges the range of applicability of line-graph routing.

The effectiveness of line-graph routing may be improved further by leveraging the freedom of which qubit is swapped with the mediating qubit. In line-graph routing, for the implementation of the gate U_{ij}^ℓ , it is qubit i that is swapped with the mediating qubit [Equation (4)]. However, in some cases, it is only when swapping qubit j with the mediating qubit that a cancellation of SWAP gates

from consecutive mediated two-qubit gates MU [Equation (4), Figure 2] occurs. Thus, line-graph routing may be improved by letting the decision of which qubit to swap with the mediating qubit depend on the ensuing gates in the input quantum circuit.

Although we showed the ability of line-graph routing to outperform general-purpose methods, we did not prove it gives the optimal routing. In fact, it was shown to be suboptimal in cases where the input circuit is a circuit on the complete graph. A proof or refutation for the optimality of line-graph routing (possibly after including the improvement of the previous paragraph) would therefore require careful consideration of the allowed input circuits and performance characteristic for which optimality is considered. Such a proof may inspire even stronger methods that build on or generalize line-graph routing.

One method for studying the optimality of line-graph routing is to compare it against solvers that find the *optimal* qubit routing strategy [30, 35, 38, 50, 55]. However, as supported by data in the Supplemental Material [23], running these optimal methods for relevant system sizes of on the order of hundreds of qubits is computationally infeasible. At the same time, the optimal swapping strategy for small input circuits and small target coupling graphs gives little information about the optimal solution on large patches. It is perhaps possible to amend the optimal routing methods so that, if present, they leverage the spatial periodicity of the input circuit and the target coupling graph, which provides an interesting route for further research by itself. Also a direct comparison between line-graph qubit routing and application-specific compilation frameworks such as Paulihedral [28] or Tetris [22] is not meaningful as the latter two achieve a task of which qubit routing is a relatively unimportant subroutine.

Error rates in quantum devices typically vary over time, from qubit to qubit (for single-qubit gates), and from qubit pair to qubit pair (for two-qubit gates). Noise-aware routing strategies take this noise variability into account, preferring the use of high fidelity qubits and qubit pairs over low fidelity qubits and qubit pairs [37, 40]. In the last step of line-graph routing, the routed circuit must be placed on a patch of the hardware coupling graph (Figure 3). This step can be made noise-aware by selecting the best patch of the hardware coupling graph according to some selection criterion, like the highest average two-gate fidelity. However, apart from this placement step, line-graph routing is noise-unaware. It is an open question whether the depth reduction offered by line-graph routing outweighs its limited noise awareness. To answer this question either requires a comparison between line-graph routing and noise-aware routing strategies on a case-by-case basis, or a broader and extensive study taking into account various levels of noise variability, input circuits and target coupling graphs, which is outside the scope of the current article.

A straightforward but exciting way to carry on the work in the current article is to run our routed circuits on actual quantum hardware. The circuits available in the Supplemental Material [23] can be executed as-is on hardware with the appropriate hardware coupling graphs. Remaining challenges therein are the extraction of useful physical quantities from the generated quantum states. To obtain results that go beyond anything that can be obtained classically requires further improvements of error mitigation techniques and quantum hardware.

All code and data used to generate the results in this article are available as Supplemental Material and at Reference [23].

References

- [1] Aniruddha Bapat, Andrew M. Childs, Alexey V. Gorshkov, and Eddie Schoute. 2023. Advantages and limitations of quantum routing. *PRX Quantum* 4, 1 (Feb 2023), 010313. DOI : <https://doi.org/10.1103/PRXQuantum.4.010313>
- [2] L. W. Beineke. 1970. Characterizations of derived graphs. *Journal of Combinatorial Theory* 9, 2 (1970), 129–135. DOI : [https://doi.org/10.1016/S0021-9800\(70\)80019-9](https://doi.org/10.1016/S0021-9800(70)80019-9)
- [3] Lowell W. Beineke and Jay S. Bagga. 2021. *Line Graphs and Line Digraphs*. Springer.

- [4] R. F. Bishop, P. H. Y. Li, D. J. J. Farnell, J. Richter, and C. E. Campbell. 2012. Frustrated Heisenberg antiferromagnet on the checkerboard lattice: J_1 - J_2 model. *Physical Review B* 85, 20 (May 2012), 205122. DOI : <https://doi.org/10.1103/PhysRevB.85.205122>
- [5] Adi Botea, Akihiro Kishimoto, and Radu Marinescu. 2018. On the complexity of quantum circuit compilation. In *Proceedings of the International Symposium on Combinatorial Search*, Vol. 9. 138–142.
- [6] Sergey Bravyi, Oliver Dial, Jay M. Gambetta, Dario Gil, and Zaira Nazario. 2022. The future of quantum computing with superconducting qubits. *Journal of Applied Physics* 132, 16 (Oct 2022), 160902. DOI : <https://doi.org/10.1063/5.0082975>
- [7] Markus Brink, Jerry M. Chow, Jared Hertzberg, Easwar Magesan, and Sami Rosenblatt. 2018. Device challenges for near term superconducting quantum processors: Frequency collisions. In *Proceedings of the 2018 IEEE International Electron Devices Meeting (IEDM'18)*, 6.1.1–6.1.3. DOI : <https://doi.org/10.1109/IEDM.2018.8614500>
- [8] Guido Burkard. 2025. Recipes for the digital quantum simulation of lattice spin systems. *SciPost Phys. Core* 8 (2025), 030. DOI : <https://doi.org/10.21468/SciPostPhysCore.8.1.030>
- [9] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. 2021. Variational quantum algorithms. *Nature Reviews Physics* 3, 9 (2021), 625–644. DOI : <https://doi.org/10.1038/s42254-021-00348-9>
- [10] Gouri Chakraborty, In-Hyeok Park, Raghavender Medishetty, and Jagadees J. Vittal. 2021. Two-dimensional metal-organic framework materials: Synthesis, structures, properties and applications. *Chemical Reviews* 121, 7 (2021), 3751–3891. DOI : <https://doi.org/10.1021/acs.chemrev.0c01049>
- [11] Christopher Chamberland, Guanyu Zhu, Theodore J. Yoder, Jared B. Hertzberg, and Andrew W. Cross. 2020. Topological and subsystem codes on low-degree graphs with flag qubits. *Physical Review X* 10, 1 (Jan 2020), 011022. DOI : <https://doi.org/10.1103/PhysRevX.10.011022>
- [12] Andrew M. Childs, Eddie Schoute, and Cem M. Unsal. 2019. Circuit transformations for quantum architectures. In *Proceedings of the 14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC'19)*. Leibniz International Proceedings in Informatics (LIPIcs), Vol. 135, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 3:1–3:24. DOI : <https://doi.org/10.4230/LIPIcs.TQC.2019.3>
- [13] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. 2004. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26, 10 (2004), 1367–1372. DOI : <https://doi.org/10.1109/TPAMI.2004.75>
- [14] Alexander Cowtan, Silas Dilkies, Ross Duncan, Alexandre Krajenbrink, Will Simmons, and Seyon Sivarajah. 2019. On the qubit routing problem. In *Proceedings of the 14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC'19)*. Leibniz International Proceedings in Informatics (LIPIcs), Vol. 135, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 5:1–5:32. DOI : <https://doi.org/10.4230/LIPIcs.TQC.2019.5>
- [15] Richard P. Feynman. 1982. Simulating physics with computers. *International Journal of Theoretical Physics* 21, 6–7 (Jun 1982), 467–488. DOI : <https://doi.org/10.1007%2Fb02650179>
- [16] J.-B. Fouet, M. Mambrini, P. Sindzingre, and C. Lhuillier. 2003. Planar pyrochlore: A valence-bond crystal. *Physical Review B* 67, 5 (Feb 2003), 054411. DOI : <https://doi.org/10.1103/PhysRevB.67.054411>
- [17] Daniel Stilck França and Raul García-Patrón. 2021. Limitations of optimization algorithms on noisy quantum devices. *Nature Physics* 17, 11 (Oct 2021), 1221–1227. DOI : <https://doi.org/10.1038/s41567-021-01356-3>
- [18] Masayoshi Fujihala, Katsuhiko Morita, Richard Mole, Setsuo Mitsuda, Takami Tohyama, Shin-ichiro Yano, Dehong Yu, Shigetoshi Sota, Tomohiko Kuwai, Akihiro Koda, et al. 2020. Gapless spin liquid in a square-kagome lattice antiferromagnet. *Nature Communications* 11, 1 (09 Jul 2020), 3429. DOI : <https://doi.org/10.1038/s41467-020-17235-z>
- [19] Frank Harary. 1969. *Graph Theory*. Addison-Wesley.
- [20] Jared B. Hertzberg, Eric J. Zhang, Sami Rosenblatt, Easwar Magesan, John A. Smolin, Jeng-Bang Yau, Vivekananda P. Adiga, Martin Sandberg, Markus Brink, Jerry M. Chow, et al. 2021. Laser-annealing Josephson junctions for yielding scaled-up superconducting quantum processors. *npj Quantum Information* 7, 1 (Aug 2021), 129. DOI : <https://doi.org/10.1038/s41534-021-00464-5>
- [21] T. Ito, N. Kakimura, N. Kamiyama, Y. Kobayashi, and Y. Okamoto. 2023. Algorithmic theory of qubit routing. In *Algorithms and Data Structures WADS 2023, Lecture Notes in Computer Science*, P. Morin and S. Suri (Eds.). Vol 14079, Springer, Cham. https://doi.org/10.1007/978-3-031-38906-1_35
- [22] Yuwei Jin, Zirui Li, Fei Hua, Tianyi Hao, Huiyang Zhou, Yipeng Huang, and Eddy Z. Zhang. 2024. Tetris: A compilation framework for VQA applications in quantum computing . In *Proceedings of the 2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA'24)*. IEEE Computer Society, Los Alamitos, CA, USA, 277–292. DOI : <https://doi.org/10.1109/ISCA59077.2024.00029>
- [23] Joris Kattemölle and Seenivasan Hariharan. 2023. Line-graph Routing (LIGRAR). Retrieved May 14th, 2025 from <https://github.com/kattemolle/LIGRAR>

- [24] Joris Kattemölle and Jasper van Wezel. 2022. Variational quantum eigensolver for the Heisenberg antiferromagnet on the kagome lattice. *Physical Review B* 106, 21 (Dec 2022), 214429. DOI : <https://doi.org/10.1103/PhysRevB.106.214429>
- [25] Olivia Lanes and A. J. Rasmussen. 2023. IBM Quantum’s Open Science Prize Returns. (Jan 2023). Retrieved May 14th, 2025 from <https://research.ibm.com/blog/ibm-quantum-open-science-prize-2022>
- [26] Andreas M. Läuchli, Julien Sudan, and Roderich Moessner. 2019. $S = \frac{1}{2}$ kagome Heisenberg antiferromagnet revisited. *Physical Review B* 100, 15 (Oct 2019), 155142. DOI : <https://doi.org/10.1103/PhysRevB.100.155142>
- [27] Gushu Li, Yufei Ding, and Yuan Xie. 2019. Tackling the qubit mapping problem for NISQ-era quantum devices. In *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS’19)*. ACM, New York, NY, USA, 1001–1014. DOI : <https://doi.org/10.1145/3297858.3304023>
- [28] Gushu Li, Anbang Wu, Yunong Shi, Ali Javadi-Abhari, Yufei Ding, and Yuan Xie. 2022. Paulihedral: A generalized block-wise compiler optimization framework for Quantum simulation kernels. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS’22)*. ACM, New York, NY, USA, 554–569. DOI : <https://doi.org/10.1145/3503222.3507715>
- [29] Elliott H. Lieb. 1989. Two theorems on the Hubbard model. *Physical Review Letters* 62, 10 (Mar 1989), 1201–1204. DOI : <https://doi.org/10.1103/PhysRevLett.62.1201>
- [30] Wan-Hsuan Lin, Jason Kimko, Bochen Tan, Nikolaj Björner, and Jason Cong. 2023. Scalable optimal layout synthesis for NISQ quantum processors. In *Proceedings of the 2023 60th ACM/IEEE Design Automation Conference (DAC’23)*. 1–6. DOI : <https://doi.org/10.1109/DAC56929.2023.10247760>
- [31] Seth Lloyd. 1996. Universal quantum simulators. *Science* 273, 5278 (Aug 1996), 1073–1078. DOI : <https://doi.org/10.1126/Science.273.5278.1073>
- [32] Simon Martiel and Timothée Goubault de Brugière. 2022. Architecture aware compilation of quantum circuits via lazy synthesis. *Quantum* 6 (June 2022), 729. DOI : <https://doi.org/10.22331/q-2022-06-07-729>
- [33] Dmitri Maslov, Sean M. Falconer, and Michele Mosca. 2008. Quantum circuit placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 4 (2008), 752–763. DOI : <https://doi.org/10.1109/TCAD.2008.917562>
- [34] Jarrod R. McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. 2016. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics* 18, 2 (Feb 2016), 023023. DOI : <https://doi.org/10.1088/1367-2630/18/2/023023>
- [35] Abtin Molavi, Amanda Xu, Martin Diges, Lauren Pick, Swamit Tannu, and Aws Albarghouthi. 2022. Qubit mapping and routing via MaxSAT. In *Proceedings of the 2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO’22)*. 1078–1091. DOI : <https://doi.org/10.1109/MICRO56248.2022.00077>
- [36] Ashley Montanaro. 2016. Quantum algorithms: An overview. *npj Quantum Information* 2, 1 (2016), 15023. DOI : <https://doi.org/10.1038/npjqi.2015.23>
- [37] Prakash Murali, Jonathan M. Baker, Ali Javadi-Abhari, Frederic T. Chong, and Margaret Martonosi. 2019. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS’19)*. ACM, New York, NY, USA, 1015–1029. DOI : <https://doi.org/10.1145/3297858.3304075>
- [38] Giacomo Nannicini, Lev S. Bishop, Oktay Günlük, and Petar Jurcevic. 2022. Optimal qubit assignment and routing via integer programming. *ACM Transactions on Quantum Computing* 4, 1, Article 7 (Oct 2022), 31 pages. DOI : <https://doi.org/10.1145/3544563>
- [39] Paul Nation, Hanhee Paik, Andrew Cross, and Zaira Nazario. 2022. The IBM Quantum Heavy Hex Lattice. (May 2022). Retrieved March 16, 2023 from <https://research.ibm.com/blog/heavy-hex-lattice>
- [40] Shin Nishio, Yulu Pan, Takahiko Satoh, Hideharu Amano, and Rodney Van Meter. 2020. Extracting success from IBM’s 20-Qubit machines using error-aware compilation. *Journal on Emerging Technologies in Computing Systems* 16, 3, Article 32 (May 2020), 25 pages. DOI : <https://doi.org/10.1145/3386162>
- [41] Daniel G. Nocera, Bart M. Bartlett, Daniel Grohol, Dimitris Papoutsakis, and Matthew P. Shores. 2004. Spin frustration in 2D Kagome lattices: A problem for inorganic synthetic chemistry. *Chemistry - A European Journal* 10, 16 (2004), 3850–3859. DOI : <https://doi.org/10.1002/chem.200306074>
- [42] M. R. Norman. 2016. Colloquium: Herbertsmithite and the search for the quantum spin liquid. *Reviews of Modern Physics* 88, 4 (Dec 2016), 041002. DOI : <https://doi.org/10.1103/RevModPhys.88.041002>
- [43] G. Parisi. 1979. Infinite number of order parameters for spin-glasses. *Physical Review Letters* 43, 23 (Dec 1979), 1754–1756. DOI : <https://doi.org/10.1103/PhysRevLett.43.1754>
- [44] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O’Brien. 2014. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications* 5, 1 (23 Jul 2014), 4213. DOI : <https://doi.org/10.1038/ncomms5213>
- [45] John Preskill. 2018. Quantum Computing in the NISQ era and beyond. *Quantum* 2 (August 2018), 79. DOI : <https://doi.org/10.22331/q-2018-08-06-79>

- [46] Qiskit contributors. 2023. Qiskit: An Open-source Framework for Quantum Computing. (2023). DOI : <https://doi.org/10.5281/zenodo.2573505>
- [47] Nicholas D. Roussopoulos. 1973. A max $\{m, n\}$ algorithm for determining the graph H from its line graph G . *Information Processing Letters* 2, 4 (1973), 108–112. DOI : [https://doi.org/10.1016/0020-0190\(73\)90029-X](https://doi.org/10.1016/0020-0190(73)90029-X)
- [48] Mohan Sarovar, Timothy Proctor, Kenneth Rudinger, Kevin Young, Erik Nielsen, and Robin Blume-Kohout. 2020. Detecting crosstalk errors in quantum information processors. *Quantum* 4 (September 2020), 321. DOI : <https://doi.org/10.22331/q-2020-09-11-321>
- [49] Lucile Savary and Leon Balents. 2016. Quantum spin liquids: A review. *Reports on Progress in Physics* 80, 1 (Nov 2016), 016502. DOI : <https://doi.org/10.1088/0034-4885/80/1/016502>
- [50] Irfansha Shaik and Jaco van de Pol. 2024. Optimal layout synthesis for deep quantum circuits on NISQ processors with 100+ qubits. arXiv:2403.11598. Retrieved from <https://arxiv.org/abs/2403.11598>
- [51] David Sherrington and Scott Kirkpatrick. 1975. Solvable model of a spin-glass. *Physical Review Letters* 35, 26 (Dec 1975), 1792–1796. DOI : <https://doi.org/10.1103/PhysRevLett.35.1792>
- [52] Rahul Siddharthan and Antoine Georges. 2001. Square kagome quantum antiferromagnet and the eight-vertex model. *Physical Review B* 65, 1 (Dec 2001), 014417. DOI : <https://doi.org/10.1103/PhysRevB.65.014417>
- [53] R. R. P. Singh, O. A. Starykh, and P. J. Freitas. 1998. A new paradigm for two-dimensional spin liquids. *Journal of Applied Physics* 83, 11 (Jun 1998), 7387–7389. DOI : <https://doi.org/10.1063/1.367682>
- [54] Marcos Yukio Siraichi, Vinicius Fernandes dos Santos, Caroline Collange, and Fernando Magno Quintão Pereira. 2019. Qubit allocation as a combination of subgraph isomorphism and token swapping. *Proceedings of the ACM on Programming Languages* 3, OOPSLA (Oct 2019), 1–29. DOI : <https://doi.org/10.1145/3360546>
- [55] Bochen Tan and Jason Cong. 2020. Optimal layout synthesis for quantum computing. In *Proceedings of the 39th International Conference on Computer-Aided Design (ICCAD'20)*. ACM, New York, NY, USA, Article 137, 9 pages. DOI : <https://doi.org/10.1145/3400302.3415620>
- [56] Bochen Tan and Jason Cong. 2020. Optimality study of existing quantum computing layout synthesis tools. *IEEE Transactions on Computers* 70, 9 (2020), 1363–1373.
- [57] Dave Wecker, Matthew B. Hastings, and Matthias Troyer. 2015. Progress towards practical quantum variational algorithms. *Physical Review A* 92, 4 (2015), 042303. DOI : <https://doi.org/10.1103/physreva.92.042303>
- [58] Robert Wille, Lukas Burgholzer, and Alwin Zulehner. 2019. Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations. In *Proceedings of the 56th Annual Design Automation Conference 2019 (DAC'19)*. ACM, New York, NY, USA, Article 142, 6 pages. DOI : <https://doi.org/10.1145/3316781.3317859>
- [59] Dian Wu, Riccardo Rossi, Filippo Vicentini, Nikita Astrakhantsev, Federico Becca, Xiaodong Cao, Juan Carrasquilla, Francesco Ferrari, Antoine Georges, Mohamed Hibat-Allah, Masatoshi Imada, Andreas M. Läuchli, Guglielmo Mazzola, Antonio Mezzacapo, Andrew Millis, Javier Robledo Moreno, Titus Neupert, Yusuke Nomura, Jannes Nys, Olivier Parcollet, Rico Pohle, Imelda Romero, Michael Schmid, J. Maxwell Silvester, Sandro Sorella, Luca F. Tocchio, Lei Wang, Steven R. White, Alexander Wietek, Qi Yang, Yiqi Yang, Shiwei Zhang, and Giuseppe Carleo. 2024. Variational benchmarks for quantum many-body problems. *Science* 386, 6719 (2024), 296–301. DOI : <https://doi.org/10.1126/science.adg9774>

Appendix

A Removal of Lone Leaf Nodes

In some cases, line-graph routing produces circuits containing qubits i that are swapped with only one corresponding mediating qubit m_i during the course of the entire circuit. In these cases, the mediating qubits m_i can be removed and replaced by the corresponding qubits i , leading to a reduction in the qubit and SWAP count of the routed circuit.

We define a *lone leaf* i as a node in a coupling graph $\text{heavy}(G)$ that is of degree one and shares its only edge with a node that has no other neighbors of degree one. We do not assume the coupling graph is a tree. As an example, consider the triangle graph with an extra node i connected to one of the nodes m_i of the triangle. The node i is a lone leaf.

If a node i is a lone leaf in a coupling graph $\text{heavy}(G)$, by the construction of line-graph routing its neighbor m must be a mediating node. For every two-qubit gate that is performed between qubit i and any other neighbor j of m , qubits i and m first need to be swapped. So, as far as the interactions with qubit i are concerned, qubit m may be fully eliminated by simply removing qubit m and reverting mediated two-qubit gates MU_{imj} to the original gates U_{ij} . Physically, qubit i can

be put on the place of qubit m . After the removal of m and the relocation of qubit i , inspection of Equation (4) reveals that qubit i may in fact act as a mediating node for any two-qubit gates between any two neighbors j, k of m unequal to i ; after a mediated gate MU_{jik} , qubit i returns to its starting position unaffected.

Received 29 February 2024; revised 25 October 2024; accepted 14 April 2025