Minimizing the number of edges in LC-equivalent graph states

Hemant Sharma,[1, 2, *] Kenneth Goodenough,[3] Johannes Borregaard,[4] Filip Rozpędek,[3] and Jonas Helsen[2]

[1]*QuTech, Delft University of Technology, 2628 CJ, Delft, The Netherlands*

[2]*QuSoft and CWI, Science Park 123, 1098 XG Amsterdam, The Netherlands*

[3]*College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, Massachusetts 01003, USA*

[4]*Department of Physics, Harvard University, Cambridge, Massachusetts 02138, USA*

(Dated: June 13, 2025)

Graph states are a powerful class of entangled states with numerous applications in quantum communication and quantum computation. Local Clifford (LC) operations that map one graph state to another can alter the structure of the corresponding graphs, including changing the number of edges. Here, we tackle the associated edge-minimization problem: finding graphs with the minimum number of edges in the LC-equivalence class of a given graph. Such graphs are called minimum edge representatives (MER), and are crucial for minimizing the resources required to create a graph state. We leverage Bouchet's algebraic formulation of LC-equivalence to encode the edge-minimization problem as an integer linear program (`ILP`). We further propose a simulated annealing (`SA`) approach guided by the local clustering coefficient for edge minimization. We identify new MERs for graph states with up to 16 qubits by combining `SA` and `ILP`. We extend the ILP to weighted-edge minimization, where each edge has an associated weight, and prove that this problem is NP-complete. Finally, we employ our tools to minimize resources required to create all-photonic generalized repeater graph states using fusion operations.

I. INTRODUCTION

Graph states are a powerful yet tractable class of multipartite entangled states that have been extensively studied in quantum science. Beyond their theoretical significance [1, 2], they support a wide range of applications, including measurement-based quantum computation [3–5], all-photonic quantum repeaters [6–11], loss-tolerant error-correcting codes [12–14], and quantum metrology [15]. Significant efforts have been directed toward both their experimental realization [16–25] and theoretical optimization, encompassing photonic graph state generation [26–30], general graph state construction [31], and state distribution [32].

A graph state $|G\rangle$ is represented by a graph G with vertices that correspond to qubits initialized in the $|+\rangle$ state, and edges that correspond to controlled-Z (CZ) gates applied to pairs of qubits. Graph states can also be defined by Pauli stabilizers, and thus constitute a subset of the stabilizer states. By applying local Clifford (LC) gates, i.e. single-qubit Clifford unitaries, one can transform a graph state $|G\rangle$ into a different *local Clifford equivalent* (or LC-equivalent) graph state $|H\rangle$ with possibly fewer edges in its graph. It was shown in Ref. [33] that two graph states are LC-equivalent if and only if their graphs are related by a sequence of graph operations called *local complementations*.

A natural goal is to identify LC-equivalent graph states with the fewest edges—referred to as *minimum edge representatives* (MERs). These serve as canonical representatives of *LC-orbits*, the equivalence classes under local complementation [34]. MERs have been studied as a means to reduce the physical resources needed for graph state preparation [31]. However, identifying MERs is challenging due to the convoluted structure of LC-orbits, as shown in Ref [34]. Existing methods typically rely on brute-force enumeration of the entire orbit via local complementations, which quickly becomes intractable since the number of orbit elements can grow exponentially with the number of qubits. As a result, prior work on MERs has been limited to graphs with up to 12 qubits [31, 34].

In this work, we introduce three algorithms to address the edge-minimization problem: given a graph G, find an LC-equivalent graph H with the minimum number of edges. Our first algorithm, `SA`, uses simulated annealing [35] to obtain approximate solutions for graphs with up to 100 qubits. The second algorithm, `ILP`, is an integer linear program based on Bouchet's algebraic characterization of LC-equivalence [36]. While `ILP` guarantees globally optimal solutions, its runtime scales exponentially in the worst case. Combining both approaches, our third method, `SA+ILP`, uses `SA` to precondition the input to `ILP`, yielding a significant constant-factor speedup. This hybrid approach enables us to compute exact MERs for graphs with up to 16 qubits. Moreover, we extend `ILP` to tackle the weighted-edge minimization problem: finding LC-equivalent graphs that minimize the total edge weight. We prove this problem is NP-complete via a reduction from the vertex-minor problem [37].

To demonstrate the practical utility of our approach, we use `SA` to optimize the resources required for the creation of all-photonic *generalized* repeater graph states (gRGS) [11]. These states can be utilized for sharing multiple units of entanglement in a quantum network. Moreover, they can be constructed using single-photon sources and fusion operations [38]. However, the probabilistic nature of fusion and photon loss leads to a high number of required single photons

* h.sharma-1@tudelft.nl

and fusion operations, with the fusion order critically affecting generation efficiency. To reduce resource requirements, we identify a subgraph of the gRGS and construct its corresponding MER, which contains fewer edges and is therefore more efficient to create. LC gates are applied at the start of the protocol to transform the MER into the desired subgraph, allowing for an optimized fusion order. Using the tools from Ref. [28], we quantify the resources needed and determine the optimal fusion sequence. Our method achieves more than an order-of-magnitude reduction in resource usage under realistic high-loss conditions.

In Section II, we provide a description of `SA` and `ILP` algorithms we use to find MERs. In Section III, we study the performance of our algorithms. Next, in Section IV, we demonstrate the applications of our framework for edge minimization. We conclude with an outlook for future work in Section V.

II. ALGORITHMS FOR EDGE MINIMIZATION

A. Simulated annealing

Here we discuss a heuristic approach for edge minimization based on simulated annealing (SA) [35]. Due to its flexibility, simulated annealing has been widely applied in various domains, including the distribution of graph states in networks [32].

SA is an iterative algorithm that runs for a given number of iterations ($k_{\max}$) while trying to approximate the optimal solution. At each iteration k, SA may move from a state s_k to another potential 'neighboring' state s_{pot} with a certain *acceptance probability*. In our implementation, states correspond to graphs, and the state s_{pot} is sampled from a set of neighboring states $S(s_k)$. Here, $S(s_k)$ denotes the subset of graph states that differ from the graph state s_k by one *local complementation*. A local complementation L_v is an operation on a graph G that acts on a vertex v by complementing the edges in the neighborhood of v, as shown in Fig. 1. We construct the set $S(s_k)$ based on a metric called the local clustering coefficient [39], and the degree D_v of a vertex v. Details for finding $S(s_k)$ are discussed in Appendix C.

Each state has an *energy* associated with it, which reflects its quality, with lower energy states considered to be better candidate solutions. The energy of a state is defined as the number of edges in the graph. The acceptance probability of SA is controlled using the *temperature* parameter ($T(k)$). Transitions from a low- to a high-energy state are more likely at higher temperatures. Thus, starting from a high temperature and slowly cooling down following a defined *temperature schedule*, SA initially explores a wide range of states and gradually converges to a state close to the ground state (state with minimum energy).

For efficient SA, the temperature schedule, i.e., the temperature $T(k)$ at each iteration k, must be properly defined. We consider a logarithmic cooling schedule [40]: at the k'th iteration, the temperature is given by $T(k) = T(1)/\log_2(k+1)$, for a given initial temperature $T(1)$. In principle, the initial temperature can be tailored for specific problem instances; however, we found $T(1) = 50$ and $T(1) = 100$ to be good enough in practice to find approximate MERs.

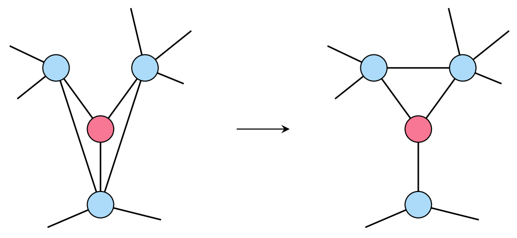


FIG. 1: Local complementation on the red vertex removes (or adds) an edge between the neighbors of the vertex if they were connected (or disconnected) prior to its application

B. Edge minimization using Bouchet's algorithm

We leverage Bouchet's LC-equivalence between graph states to find the exact solutions to the edge-minimization problem. By definition, the edge-minimization problem can be written as follows:

$$\begin{aligned} \min_{H} \quad & \sum_{i>j}^{n} A_H[i,j], \\ \text{s.t.} \quad & G \equiv_{\text{LC}} H, \end{aligned}$$

where A_G (A_H) represents the adjacency matrix of G (H). We use Bouchet's algorithm for LC-equivalence [36] to express LC-equivalence in a form amenable to (integer) linear programming. For completeness, we describe Bouchet's algorithm in Appendix A. The algorithm takes as input the adjacency matrices A_G and A_H of n-vertex graphs G and H respectively, and is based on the binary/symplectic framework (see for example [41, 42] and Appendix A for further information) to find the following equation describing LC-equivalence of graphs G and H,

$$A_G P A_H + A_G Q + A_H R + S = 0 \bmod 2 \,, \tag{1}$$
$$p_i s_i + r_i q_i = 1 \,. \tag{2}$$

Here, p_i, q_i, r_i, s_i are binary entries of $n \times n$ diagonal matrices P, Q, R, S, respectively. These matrices P, Q, R, S arise in the binary framework of LC operations, where local Clifford operations can be represented by non-singular $2n \times 2n$ matrices M

$$M = \begin{bmatrix} P & Q \\ R & S \end{bmatrix}, \tag{3}$$

such that the condition in Eq. (2) corresponds to the symplecticity. We express the LC-equivalence of G and H by setting Eq. (1) and (2) as constraints. Furthermore, we convert Eq. (1)

from a binary constraint to a integer one, by introducing a matrix B of integer variables such that:

$$A_G P A_H + A_G Q + A_H R + S = 2B. \quad (4)$$

We then linearize the equations by introducing constraints to convert products of integer variables into linear systems. This is done for products of variables originating from the terms $A_H P$ and $A_H R$ in Eq. (4) and terms PS and RQ in Eq. (2). The approach outlined above produces the full set of constraints; since these are rather unwieldy, we defer the complete derivation of the `ILP` algorithm to Appendix B.

The runtime of the `ILP` algorithm depends on the number of constraints, which in turn depend on the number of vertices and edges in the input graphs. Therefore, we employ `SA` to find the approximate MER of a given graph before inputting it into `ILP`. This preprocessing step reduces the number of constraints in the corresponding `ILP` thus improving the runtime. We refer to this combined algorithm as `SA+ILP`. We use the MOSEK optimizer API for Python [43] to run `ILP` and `SA+ILP`. Since `ILP` is an exact optimization, the results obtained are global minima, i.e. MERs of their corresponding LC-orbit. In Section III, we benchmark the performance of our algorithms for different graph models.

C. Weighted-edge minimization

Our formulation of Bouchet's algorithm as an optimization problem can be easily extended to include weighted-edge minimization of LC-equivalent graph states. The optimization problem then becomes

$$\begin{aligned} \min_{H} \quad & \sum_{i>j}^{n} A_H[i,j] W[i,j], \\ \text{s.t.} \quad & G \equiv_{\text{LC}} H \end{aligned}$$

where W is a weighted adjacency matrix encoding the weights $W[i,j]$ of each edge (i,j). These weights could represent, for example, the cost of creating edges, thereby enabling a more realistic analysis of the resource requirements for generating graph states, similar to the simulated annealing based approach of Ref. [32].

The decision problem for arbitrary weights can be proved to be NP-complete by reducing the so-called *vertex-minor problem* to the weighted edge-minimization problem. The vertex-minor (or qubit-minor) problem plays an important role in quantum information theory [44, 45]. A graph state $|H\rangle$ that can be obtained from $|G\rangle$ under local Clifford unitaries and Pauli measurements is called a qubit-minor of $|G\rangle$ [44, 45]. Accordingly, the graph H is called a vertex-minor of graph G [46]. We consider the labeled version of the vertex-minor problem, where the resultant graph after measurements and local complementations needs to be exactly H (not up to graph isomorphism). While local equivalence of two graphs G and H can be decided efficiently, deciding whether H is a vertex-minor of G is NP-complete, for both the labeled [45] and unlabeled case [37].

The vertex-minor problem can be formulated using the weighted-edge minimization problem as follows. Assume that the graph $H = (V', E')$ has no isolated vertices [47]. We define the weights $W[i,j]$ according to the following rule:

$$W[i,j] = \begin{cases} -1 & \text{if } \{i,j\} \in E' \\ +1 & \text{if } i,j \in V' \wedge \{i,j\} \notin E' \\ 0 & \text{else} \end{cases} . \quad (5)$$

The smallest weight possible is -$|E'|$, which is achievable if and only if G is LC-equivalent to a graph $\overline{G}$ such that $\overline{G}[V'] = H$, i.e. when H is a vertex-minor of G. In other words, an efficient algorithm for deciding whether the minimum score equals $-|E'|$ would also yield an efficient method for determining whether H is a vertex-minor of G, and vice versa. Since the vertex-minor problem is known to be NP-complete in general [37], it follows immediately that the decision problem for weighted-edge minimization is NP-complete as well. We note that this still leaves open whether the minimization problem remains NP-complete when restricting to uniform/non-negative weights.

III. EDGE MINIMIZATION OF LC-EQUIVALENT GRAPH STATES

In this section, we first characterize the runtime of `SA`. We then empirically evaluate: (1) the runtime of `ILP` and `SA+ILP`, and (2) the reduction in edge count achieved by `SA` and `SA+ILP`. We focus on two graph models: (1) the Erdős-Rényi model, which is a standard model for creating random graphs, and (2) the bounded-degree model, which encompasses realistic hardware constraints. Our algorithms ran on each input graph by allocating them on 20 cores of Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz, each with 32GB of memory.

We first benchmark our algorithms for the Erdős-Rényi (ER) graph model [48, 49]. An ER graph $G(n,p)$ is a random graph on n vertices, where each of the $\frac{n(n-1)}{2}$ possible edges is included independently with a probability p. The expected number of edges in the initial graph increases as the probability p increases. We can thus control the sparsity (or density) of edges in the generated graphs by changing the value of p. We first characterize the runtime of the `SA` algorithm by running it for different numbers of maximum iterations. The initial temperature is set to $T(1) = 50$ for a set of 100 Erdős-Rényi graphs with 100 vertices and 2971.2 edges on average. For $k_{\text{max}} = 50$, `SA` had an average runtime of 0.94 seconds, and the number of edges obtained in the output graphs was 2367.26 on average. Increasing k_{max} to 1050 increases the runtime to 18.82 seconds, and the number of edges obtained was 2310.73. In the rest of the benchmarking, we set the value of k_{max} and $T(1)$ to 100.

We generate graphs for increasing values of p with $n = 13$ vertices using the ER graph model. We generate 100 ER graphs for each value of p, and we apply `SA` and `SA+ILP`

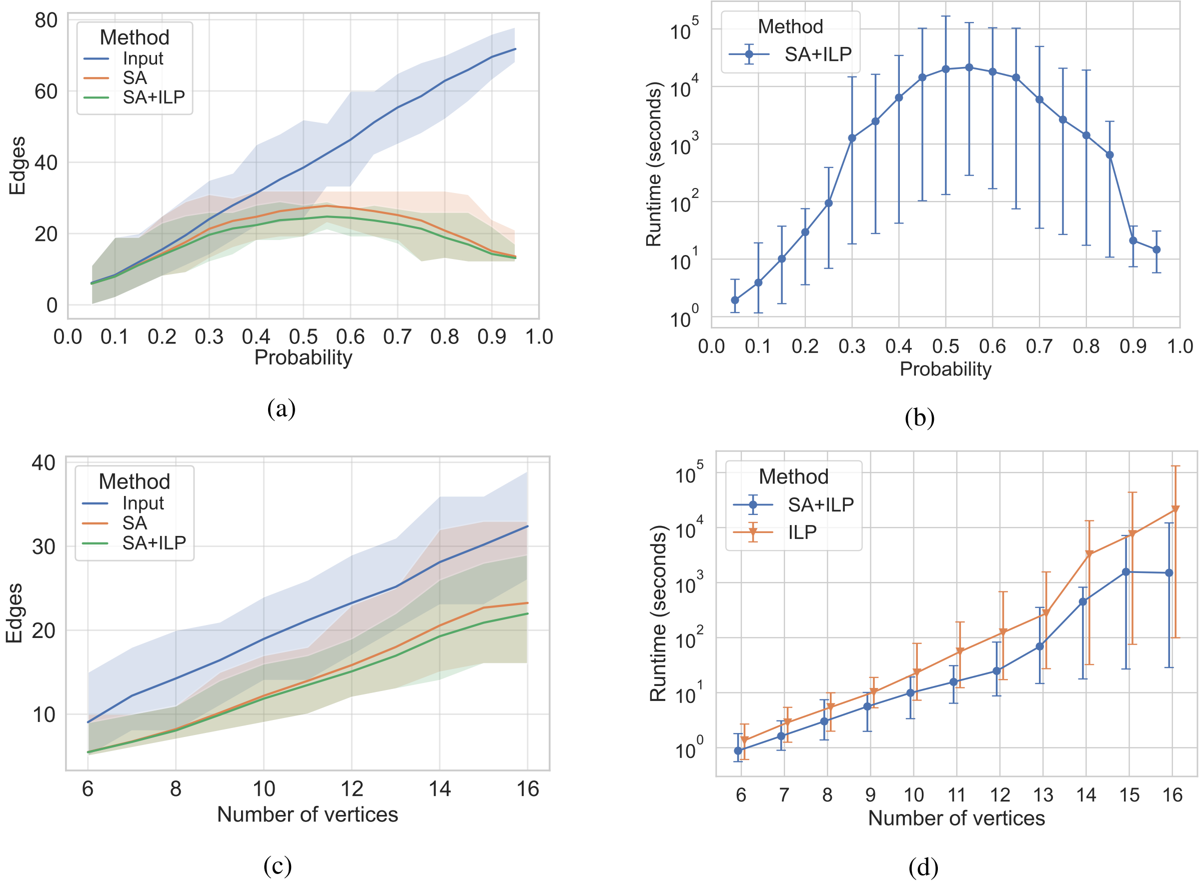
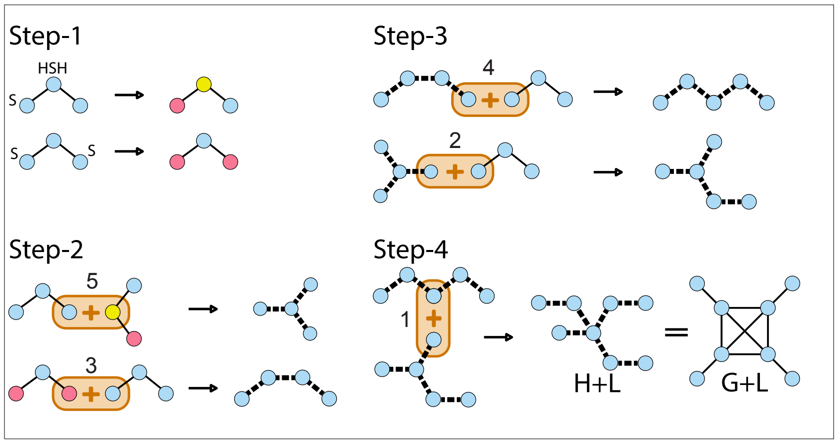


FIG. 2: In **(a)** and **(c)**, we compare the number of edges in the input graphs and the outputs from SA and SA+ILP for Erdős-Rényi graphs and bounded-degree graphs respectively. The solid lines denote the average value, and the shaded bands indicate the maximum and minimum. **(b)** shows the runtime of the SA+ILP algorithm as a function of edge inclusion probability for ER graphs with 13 vertices. **(d)** compares the runtime between ILP and SA+ILP for bounded-degree graphs. The solid line indicates the average runtime, and error bars show the 95th percentile around the mean. For ER graphs, the runtime and the MER edge count are highest around $p \sim 0.5$. From **(d)**, we observe that SA+ILP achieves a constant-factor reduction in runtime compared to ILP on average. This improvement can be attributed to the SA preprocessing step, although the runtime remains exponential in the size of the input graph. The average runtime of SA+ILP for graphs with 16 vertices is smaller than that for graphs with 15 vertices, which we attribute to a better SA performance.

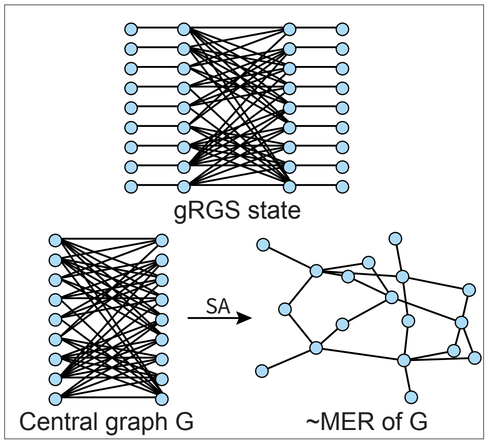
to minimize the edges in each graph generated for a p value. From Fig. 2(a), we find that the number of edges obtained from SA is on average 1.07 times larger than the MERs for ER graphs. The expected number of edges is concave in p, with a maximum at around $p \sim \frac{1}{2}$, and SA approximations are slightly worse near this regime. Moreover, we observe that dense graphs ($p \geq 0.6$) are LC-equivalent to graphs with relatively fewer edges, shown by the large gap between the input edge count and MER edge count. In Fig. 2(b), the correlation coefficient between the logarithm of runtime of SA+ILP and the number of input edges is 0.94, confirming an exponential runtime dependence on the size of input graph. Thus, we omit running the ILP algorithm for ER graphs in Fig. 2(b) since the number of edges grows to a large number ($\sim$ 70) for higher values of probability p.

We also benchmark our algorithms using the bounded-degree graph model, which incorporates hardware constraints that caps the maximum degree of edges like restricted qubit connectivity. We generate random graphs with n vertices and bounded degree of d_{lim} by sampling a set of n positive integers where all elements are $\leq d_{\text{lim}}$ and constructing simple connected graphs using the Havel-Hakimi algorithm [50, 51]. In our simulations, we fixed the value of d_{lim} to 5 and generated graphs with n ranging from 6 to 16. For each vertex number, we generated 100 graphs; however, only 63 such graphs could be generated for $n = 6$ vertices.

In Fig. 2(c), we benchmark the edge minimization performance of SA against the MERs obtained from SA+ILP for bounded-degree graphs. The number of edges obtained from SA is on average 1.04 times larger than the MERs. We observe that the performance of SA declines slightly as the number of vertices increases, likely due to the increased sparsity



(a)



(b)

FIG. 3: **(a)** demonstrates the fusion order for creating an RGS with eight qubits, obtained using `OptGraphState` from Ref. [28]. The order corresponds to the graph H+L with LC-gates applied at the beginning. Graph states (shown with blue vertices and solid edges) may transform into non-graph states when acted upon by LC-gates (represented with red and yellow vertices). The resulting state might also be a non-graph state if such a qubit is involved in fusions, indicated using dashed edges. **(b)** shows: (1) the sampled gRGS state capable of sharing three ebits across a quantum network, (2) the central graph G (with 46 edges), and (3) its approximate MER H, which contains 23 edges.

of the input graphs. In addition to edge count comparison, we compare the runtime of `ILP` and `SA+ILP` in Fig. 2(d). To find the dependence of runtime of `SA+ILP`, we calculate the correlation coefficient between the logarithm of its runtime and the number of edges and vertices input to the `ILP` part of the algorithm. The correlation coefficients between the logarithm of runtime and input edges (vertices) were 0.96 (0.89). Moreover, the addition of `SA` as a pre-processing step to `ILP` reduces the runtime of the `SA+ILP` algorithm by a factor of two compared to `ILP` alone for bounded-degree graphs. Thus, `SA+ILP` is a faster algorithm than `ILP` but it still has an exponential runtime dependence on the number of edges and vertices in the input graphs.

IV. CREATION OF GENERALIZED REPEATER GRAPH STATES

To demonstrate the utility of our algorithms, we consider the generation of photonic repeater graph states for long distance quantum communication. Repeater graph states (RGS) [6] are a key component of all-photonic quantum repeaters, enabling the distribution of a single unit of bipartite entanglement, known as an ebit over long distances. The implementation of such all-photonic repeaters has been extensively studied [26, 52–54] but often comes with a large overhead in the number of required single photon sources. A more general class of all-photonic states called *generalized* repeater graph states (gRGS) was introduced in Ref. [11], which offers a better scaling between photon overhead and the number of shared ebits. Here, we apply our MER algorithms to further minimize the number of single photons and optical *fusion* operations for the creation of gRGS.

Fusions are probabilistic operations that join two graph states into a larger graph state upon success [38]. The success probability of a standard implementation of a fusion is upper-bounded by 50%. However, in practice the success probability is often lower than 50% due to hardware losses. The fundamental resource we consider for creating graph states is a 3-qubit GHZ state (resource state). Such a state can be created probabilistically but in a heralded fashion by fusing six single photons [55].

The probabilistic nature of fusions necessitates a significant amount of resources to construct practically useful graph states. Creating all-photonic graph states involves multiple rounds of fusions starting with resource states that are fused to create intermediate graph states, which are incrementally fused to construct the final graph state. For a given graph, Ref. [28] constructs a so-called *fusion network*, indicating the order in which fusions should occur. In general, it is desirable to fuse two graph states of similar sizes to mitigate the effects caused by fusion failure. Moreover, multiple fusion networks may exist for a given graph, each potentially requiring different amounts of resources. Ref. [28] addresses these challenges and reduces the resource overhead for photonic graph state creation by optimizing (1) the number of required fusions, (2) the number of resource states, and (3) the fusion order. Moreover, they provide a numerical tool for this purpose called the `OptGraphState`.

Here, we exploit the structure of generalized repeater graph states to optimize their experimental creation. Such graphs consist of a densely-connected central graph (G) and leaves (L) that are attached to each vertex of G. We find an approximate MER (H) of the central graph G using our `SA` algorithm. Naively, one could fuse resource states to create G+L directly. However, a more resource-efficient method for creat-

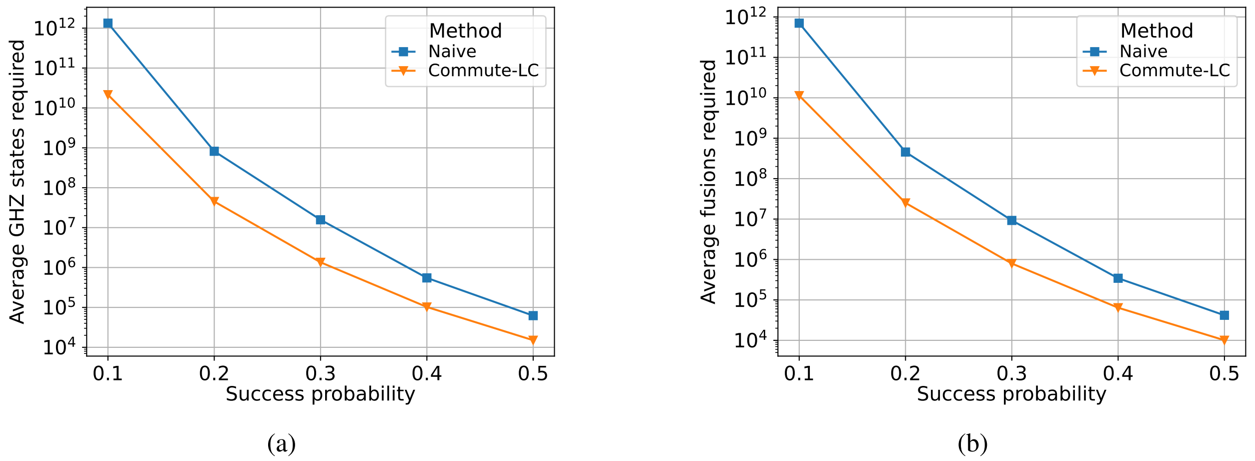


FIG. 4: **(a)** and **(b)** show the required number of resource states and fusions, respectively, for creating a single instance of G+L gRGS with 64 edges and 36 vertices (see Fig. 3(b)). We compare the naive approach of direct gRGS construction (blue curve) with the Commute-LC protocol (orange curve). The Commute-LC protocol requires nearly two orders of magnitude fewer resources than the naive approach, especially at lower fusion success probabilities. The resource savings decrease to less than an order of magnitude for relatively higher success probabilities.

ing a gRGS could be: (1) creating the graph H using fusions, (2) applying LC-gates to convert the graph H to G, and (3) adding leaves L to G. We will refer to this way of construction as *Intermediate-LC*. However, the addition of leaves at the end constrains the order of fusions which could affect the probability of creation of the gRGS. We propose commuting the LC-gates to the beginning of the protocol that lifts this constraint and allows for an optimal fusion order. We refer to this protocol as *Commute-LC*. In conclusion, the Commute-LC protocol has two steps: (1) apply the LC-gates to the resource states at the beginning of the protocol, (2) follow fusion order to create the graph H+L using fusions. The LC-gates here correspond to the conversion of the graph H to G. We should note that fusions involve the application of single-qubit Pauli correction operations, but they can be commuted through Clifford gates since these are Pauli corrections.

We illustrate the construction of G+L using the Commute-LC protocol in Fig. 3(a). Following the steps in Commute-LC, we first apply LC-gates to the appropriate qubits, followed by fusing states according to the order obtained for creating H+L. Applying LC-gates converts the resource states into LC-equivalent *non-graph* states. The Commute-LC protocol fuses these non-graph states following the order of H+L to obtain the desired graph state G+L. Appendix D details how to determine locations of LC-gates and how following the order for H+L constructs the G+L graph. We note here that Ref. [28] also applies LC-gates to their qubits during their optimization. However, these LC-gates only convert a fully-connected subgraph (clique) into a star. In contrast, by applying SA to the central graph G in Protocols 1 and 2, we generalize the use of LC-gates to simplify the fusion process not just for cliques but for arbitrary densely connected subgraphs.

In Fig. 4, we compare two methods: (1) the naive approach of creating G+L directly, and (2) creation following the Commute-LC protocol. We use the Python package OptGraphState to estimate the required resources. We consider a gRGS that can share 3 ebits across a network [56], shown in Fig. 3(b). This gRGS has a central graph G with 18 vertices and 46 edges, shown in Fig. 3(b). We find an approximate MER of this graph with 23 edges using the SA algorithm (with $k_{\text{max}} = 10000$ and $T(1) = 100$). We observe that commuting LC-gates to the beginning and following fusion order for creating H+L significantly reduces the required resources. This reduction is nearly two orders of magnitude for fusions in high-loss conditions.

V. OUTLOOK

We have proposed three algorithms to address the edge-minimization problem. First, a simulated annealing (SA) approach efficiently finds approximate MERs for graphs with up to 100 qubits. Second, we introduce an integer linear programming (ILP) formulation that yields exact solutions, albeit with potentially exponential runtime. Third, we combine both methods into a hybrid SA+ILP algorithm, enabling exact MER identification for graphs with up to 16 qubits.

We benchmark all algorithms on two graph models: bounded-degree and Erdős-Rényi. Additionally, we extend the ILP to handle weighted-edge minimization and prove that this variant is NP-complete. Importantly, we show that the scalable SA approach manages to find approximate MERs with less than 10% more edges on average than the actual MERs for a large set of randomly sampled graphs.

We have applied our SA algorithm to minimize the required resources for all-photonic generalized repeater graph states (gRGS) demonstrating its practical utility for long distance quantum communication. Specifically, we have proposed to apply LC-gates corresponding to local complementation to the resource states before fusing them to create the gRGS states. We then demonstrate that SA can be combined with existing methods from Ref. [28] to reduce the required number of single-photon sources and fusion operations be more than an order of magnitude for realistic lossy implementations.

Our results opens up a number of future directions. Finding the MER can reduce the number of required CZ gates for creating a desired graph state between qubits following the idea in Ref. [31]. However, there are examples of graph states where the minimum number of edges up to local complementations is greater than the minimum number of two-qubit gates needed to create the state [57–59]. It would be interesting to further investigate the exact connection between MERs and the minimal number of CZ-gates for generating qubit graph state generation

Furthermore, we have focused exclusively on transformations between graph states that are allowed with local Clifford unitaries. In practical scenarios, however, one might consider arbitrary local unitaries (LU), which define a strictly coarser equivalence relation on graphs than the LC-equivalence relation, as was first shown in [60]. Graph states under local unitary equivalence are significantly less well-understood, but this topic has seen recent investigations [61–63], closely relating this equivalence to the Clifford hierarchy. One natural follow-up question is whether edge minimization of graphs under local unitaries can similarly be phrased as an ILP (using, for example, the algorithm from [63]), and, if so, to what extent this coarser equivalence can reduce the edge count.

ACKNOWLEDGMENTS

We thank Tim Coopmans and Sebastiaan Brand for useful conversations, and Luise Prielinger for her help with the HPC. JH acknowledges funding from the Dutch Research Council (NWO) through a Veni grant (grant No.VI.Veni.222.331). JH and HS acknowledge funding from the Quantum Software Consortium (NWO Zwaartekracht Grant No.024.003.037). JB acknowledges support from The AWS Quantum Discovery Fund at the Harvard Quantum Initiative. Part of this work was performed while HS was on a research visit at Harvard University funded by a Quantum Delta NL travel grant.

CODE AVAILABILITY

A release of our code is publicly available at `https://doi.org/10.5281/zenodo.15534419` and the accompanying data at `https://doi.org/10.5281/zenodo.15534839`, with the expectation that it will serve as a valuable resource for the quantum information community in addressing practical optimization challenges related to graph states. The usage guidelines and notes can be found at the GitHub repository `https://github.com/arr0w-hs/graph_state_optimization`.

[1] M. Hein, J. Eisert, and H. J. Briegel, Multiparty entanglement in graph states, Physical Review A—Atomic, Molecular, and Optical Physics **69**, 062311 (2004).

[2] M. Hein, W. Dür, J. Eisert, R. Raussendorf, M. Van den Nest, and H.-J. Briegel, Entanglement in graph states and its applications, in *Quantum computers, algorithms and chaos* (IOS Press, 2006) pp. 115–218.

[3] R. Raussendorf and H. J. Briegel, A one-way quantum computer, Physical review letters **86**, 5188 (2001).

[4] R. Raussendorf, D. E. Browne, and H. J. Briegel, Measurement-based quantum computation on cluster states, Physical review A **68**, 022312 (2003).

[5] H. J. Briegel, D. E. Browne, W. Dür, R. Raussendorf, and M. Van den Nest, Measurement-based quantum computation, Nature Physics **5**, 19 (2009).

[6] K. Azuma, K. Tamaki, and H.-K. Lo, All-photonic quantum repeaters, Nature communications **6**, 1 (2015).

[7] M. Pant, H. Krovi, D. Englund, and S. Guha, Rate-distance tradeoff and resource costs for all-optical quantum repeaters, Physical Review A **95**, 012304 (2017).

[8] F. Rozpędek, K. P. Seshadreesan, P. Polakos, L. Jiang, and S. Guha, All-photonic gottesman-kitaev-preskill–qubit repeater using analog-information-assisted multiplexed entanglement ranking, Physical Review Research **5**, 043056 (2023).

[9] E. Kaur, A. Patil, and S. Guha, Resource-efficient and loss-aware photonic graph state preparation using an array of quantum emitters, and application to all-photonic quantum repeaters, arXiv preprint arXiv:2402.00731 (2024).

[10] A. Patil and S. Guha, An improved design for all-photonic quantum repeaters, arXiv preprint arXiv:2405.11768 (2024).

[11] B. Li, K. Goodenough, F. Rozpędek, and L. Jiang, Generalized quantum repeater graph states, arXiv preprint arXiv:2407.01429 (2024).

[12] J. Borregaard, H. Pichler, T. Schröder, M. D. Lukin, P. Lodahl, and A. S. Sørensen, One-way quantum repeater based on near-deterministic photon-emitter interfaces, Physical Review X **10**, 021071 (2020).

[13] T. J. Bell, L. A. Pettersson, and S. Paesani, Optimizing graph codes for measurement-based loss tolerance, PRX Quantum **4**, 020328 (2023).

[14] A. Patil and S. Guha, Tree cluster state generation using percolation, in *Quantum 2.0* (Optica Publishing Group, 2023) pp. QTh4A–3.

[15] N. Shettell and D. Markham, Graph states as a resource for quantum metrology, Physical review letters **124**, 110502 (2020).

[16] I. Schwartz, D. Cogan, E. R. Schmidgall, Y. Don, L. Gantz, O. Kenneth, N. H. Lindner, and D. Gershoni, Deterministic generation of a cluster state of entangled photons, Science **354**, 434 (2016).

[17] M. V. Larsen, X. Guo, C. R. Breum, J. S. Neergaard-Nielsen, and U. L. Andersen, Deterministic generation of a two-dimensional cluster state, Science **366**, 369 (2019).

[18] W. Asavanant, Y. Shiozawa, S. Yokoyama, B. Charoensombutamon, H. Emura, R. N. Alexander, S. Takeda, J.-i. Yoshikawa, N. C. Menicucci, H. Yonezawa, *et al.*, Generation of time-domain-multiplexed two-dimensional cluster state, Science **366**, 373 (2019).

[19] D. Istrati, Y. Pilnyak, J. Loredo, C. Antón, N. Somaschi, P. Hilaire, H. Ollivier, M. Esmann, L. Cohen, L. Vidro, *et al.*, Sequential generation of linear cluster states from a single photon emitter, Nature communications **11**, 5501 (2020).

[20] P. Thomas, L. Ruscio, O. Morin, and G. Rempe, Efficient generation of entangled multiphoton graph states from a single atom, Nature **608**, 677 (2022).

[21] C. Roh, G. Gwak, Y.-D. Yoon, and Y.-S. Ra, Generation of three-dimensional cluster entangled state, arXiv preprint arXiv:2309.05437 (2023).

[22] S. Cao, B. Wu, F. Chen, M. Gong, Y. Wu, Y. Ye, C. Zha, H. Qian, C. Ying, S. Guo, *et al.*, Generation of genuine entanglement up to 51 superconducting qubits, Nature **619**, 738 (2023).

[23] J. O'Sullivan, K. Reuer, A. Grigorev, X. Dai, A. Hernández-Antón, M. H. Muñoz-Arias, C. Hellings, A. Flasby, D. C. Zanuz, J.-C. Besse, *et al.*, Deterministic generation of a 20-qubit two-dimensional photonic cluster state, arXiv preprint arXiv:2409.06623 (2024).

[24] V. S. Ferreira, G. Kim, A. Butler, H. Pichler, and O. Painter, Deterministic generation of multidimensional photonic cluster states with a single quantum emitter, Nature Physics , 1 (2024).

[25] P. Thomas, L. Ruscio, O. Morin, and G. Rempe, Fusion of deterministically generated photonic graph states, Nature **629**, 567 (2024).

[26] B. Li, S. E. Economou, and E. Barnes, Photonic resource state generation from a minimal number of quantum emitters, npj Quantum Information **8**, 11 (2022).

[27] E. Takou, E. Barnes, and S. E. Economou, Optimization complexity and resource minimization of emitter-based photonic graph state generation protocols (2024), arXiv:2407.15777 [quant-ph].

[28] S.-H. Lee and H. Jeong, Graph-theoretical optimization of fusion-based graph state generation, Quantum **7**, 1212 (2023).

[29] Y. Li, Y. Dai, A. Pawar, R. Dong, J. Yang, Y. Zhang, and X. Tang, Using reinforcement learning to guide graph state generation for photonic quantum computers (2024), arXiv:2412.01038 [quant-ph].

[30] D. Bhatti and K. Goodenough, Distributing graph states with a photon-weaving quantum server, arXiv preprint arXiv:2504.07410 (2025).

[31] A. Cabello, L. E. Danielsen, A. J. López-Tarrida, and J. R. Portillo, Optimal preparation of graph states, Physical Review A—Atomic, Molecular, and Optical Physics **83**, 042314 (2011).

[32] T. Ji, J. Liu, and Z. Zhang, Distributing arbitrary quantum cluster states by graph transformation (2024), arXiv:2404.05537 [quant-ph].

[33] M. Van den Nest, J. Dehaene, and B. De Moor, Graphical description of the action of local clifford transformations on graph states, Phys. Rev. A **69**, 022316 (2004).

[34] J. C. Adcock, S. Morley-Short, A. Dahlberg, and J. W. Silverstone, Mapping graph state orbits under local complementation, Quantum **4**, 305 (2020).

[35] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Optimization by simulated annealing, Science **220**, 671 (1983).

[36] A. Bouchet, An efficient algorithm to recognize locally equivalent graphs, Combinatorica **11**, 315 (1991).

[37] A. Dahlberg, J. Helsen, and S. Wehner, The complexity of the vertex-minor problem, Information Processing Letters **175**, 106222 (2022).

[38] D. E. Browne and T. Rudolph, Resource-efficient linear optical quantum computation, Phys. Rev. Lett. **95**, 010501 (2005).

[39] D. J. Watts and S. H. Strogatz, Collective dynamics of 'small-world' networks 10.1038/30918 (1998).

[40] B. Hajek, Cooling schedules for optimal annealing, Mathematics of Operations Research **13**, 311 (1988).

[41] J. Dehaene, M. Van den Nest, B. De Moor, and F. Verstraete, Local permutations of products of bell states and entanglement distillation, Physical Review A **67**, 022310 (2003).

[42] J. Dehaene and B. De Moor, Clifford group, stabilizer states, and linear and quadratic operations over gf (2), Physical Review A **68**, 042318 (2003).

[43] M. ApS, *MOSEK Optimizer API for Python 10.2.1* (2024).

[44] A. Dahlberg and S. Wehner, Transforming graph states using single-qubit operations, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **376**, 20170325 (2018).

[45] A. Dahlberg, J. Helsen, and S. Wehner, How to transform graph states using single-qubit operations: computational complexity and algorithms, Quantum Science and Technology **5**, 045016 (2020).

[46] S.-i. Oum, Rank-width and vertex-minors, Journal of Combinatorial Theory, Series B **95**, 79 (2005).

[47] For the case of H containing isolated vertices, assign a weight of 0 to any of the edges incident to an isolated vertex.

[48] P. Erdős and A. Rényi, On random graphs i, Publicationes Mathematicae. 6 (3–4): 290–297 (1959).

[49] E. N. Gilbert, Random graphs, Annals of Mathematical Statistics 30, 1141 (1959).

[50] S. L. Hakimi, On realizability of a set of integers as degrees of the vertices of a linear graph. i, Journal of the Society for Industrial and Applied Mathematics **10**, 496 (1962), https://doi.org/10.1137/0110037.

[51] V. Havel, Poznámka o existenci konečných grafů, Časopis pro pěstování matematiky **080**, 477 (1955).

[52] E. Takou, E. Barnes, and S. E. Economou, Optimization complexity and resource minimization of emitter-based photonic graph state generation protocols, arXiv preprint arXiv:2407.15777 (2024).

[53] D. Buterakos, E. Barnes, and S. E. Economou, Deterministic generation of all-photonic quantum repeaters from solid-state emitters, Physical Review X **7**, 041023 (2017).

[54] Y. Zhan, P. Hilaire, E. Barnes, S. E. Economou, and S. Sun, Performance analysis of quantum repeaters enabled by deterministically generated photonic graph states, Quantum **7**, 924 (2023).

[55] M. Varnava, D. E. Browne, and T. Rudolph, How good must single photon sources and detectors be for efficient linear optical quantum computation?, Physical review letters **100**, 060502 (2008).

[56] Similar to Ref. [11].

[57] S. Kumabe, R. Mori, and Y. Yoshimura, Complexity of graph-state preparation by clifford circuits, arXiv preprint arXiv:2402.05874 (2024).

[58] A. Jena, Graph-theoretic techniques for optimizing nisq algorithms, (2024).

[59] J. Davies and A. Jena, Preparing graph states forbidding a vertex-minor, arXiv preprint arXiv:2504.00291 (2025).

[60] Z. Ji, J. Chen, Z. Wei, and M. Ying, The lu-lc conjecture is false, arXiv preprint arXiv:0709.1266 (2007).

[61] A. Burchardt, J. de Jong, and L. Vandré, Algorithm to verify local equivalence of stabilizer states, arXiv preprint arXiv:2410.03961 (2024).
[62] N. Claudet and S. Perdrix, Local equivalence of stabilizer states: a graphical characterisation, arXiv preprint arXiv:2409.20183 (2024).
[63] N. Claudet and S. Perdrix, Deciding local unitary equivalence of graph states in quasi-polynomial time, arXiv preprint arXiv:2502.06566 (2025).
[64] M. Van den Nest, J. Dehaene, and B. De Moor, Efficient algorithm to recognize the local clifford equivalence of graph states, Physical Review A—Atomic, Molecular, and Optical Physics **70**, 034302 (2004).

Appendix A: LC-equivalence algorithm for graph states

We will now briefly recap Bouchet's algorithm, using the stabilizer formalism (as was done in [64]). In particular, we will focus on the commonly used symplectic framework of the stabilizer formalism [41, 64]. In the symplectic framework, Pauli strings are mapped to elements in $\mathbb{F}_2^{2n}$ through a map ϕ. Furthermore, conjugation of Pauli strings by Clifford unitaries correspond to symplectic transformations of $\mathbb{F}_2^{2n}$. That is, for each Clifford C there exists a $2n \times 2n$ binary matrix M such that $\phi\left(CsC^\dagger\right) = M\phi\left(s\right)$, for every Pauli string s. The matrix M being symplectic means that

$$M^T \Omega M = \Omega\,, \text{ where } \Omega = \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}\,, \tag{A1}$$

and I is the $n \times n$ identity matrix. This encodes the fact that Clifford transformations need to preserve the commutation relations between Pauli strings. In fact, every matrix that satisfies the above symplectic condition corresponds to some Clifford transformation. In what follows, we will identify Pauli strings and Clifford unitaries and their images under ϕ.

The generator sets of the two graph states $|G\rangle$ and $|H\rangle$ under consideration are given by

$$\begin{bmatrix} A_G \\ I \end{bmatrix}, \begin{bmatrix} A_H \\ I \end{bmatrix}, \tag{A2}$$

where I is an $n \times n$ identity matrix, and A_G and A_H are the adjacency matrices of G and H, respectively. Indeed, the i'th column of each tableaux corresponds to the i'th stabilizer generator.

In the symplectic framework, G and H are LC-equivalent if and only if there exist M and W such that

$$M \begin{bmatrix} A_H \\ I \end{bmatrix} W = \begin{bmatrix} A_G \\ I \end{bmatrix} \tag{A3}$$

$$M = \begin{bmatrix} P & Q \\ R & S \end{bmatrix}, \; W \in \mathrm{GL}_{2n}\left(\mathbb{F}_2\right)\,. \tag{A4}$$

Here M represents single-qubit Clifford gates, and the P, Q, R, S submatrices are $n \times n$ diagonal matrices with binary entries p_i, q_i, r_i, s_i such that

$$p_i s_i + r_i q_i = 1, \tag{A5}$$

which corresponds to the symplectic constraint. The W matrix encodes the fact that the generating sets need not be equal, but should span the same space. That is, the resulting generator matrices only need to be equivalent up to invertible linear transformations.

As was noted in [64], one can get rid of the degree of freedom of W. This is since the columns of two matrices K_1, K_2 of rank n generate the same stabilizer group if and only if $K_1^T \Omega K_1 = K_1^T \Omega K_2 = K_2^T \Omega K_2$ is the zero matrix. It thus suffices to check whether there exists an M such that

$$\begin{bmatrix} A_G & I \end{bmatrix} \Omega M \begin{bmatrix} A_H \\ I \end{bmatrix} = 0\,. \tag{A6}$$

Expanding Eq. (A6) yields:

$$A_G P A_H + A_G Q + A_H R + S = 0. \tag{A7}$$

A solution vector (p_i, q_i, r_i, s_i) of the above linear system corresponds to an LC operation if and only if it follows the constraint in Eq. (A5).

Eq. (A7) is a set of n^2 equations in $4n$ variables (p_i, q_i, r_i, s_i) with a constraint (Eq. (A5)) that is non-linear, so in principle not easily solvable. However, through an ingenious argument, Bouchet [36] proved that only a small number of solutions to Eq. A7 have to be checked to verify if there exists one that also satisfies Eq. (A5), making the total runtime polynomial [36]. However for edge minimization, we will ignore this extra structure and just use Eqs. A7 and A5 to construct our ILPs.

Appendix B: Derivation of `ILP`

In this section, we show how we formulate an ILP for edge minimization inspired by Bouchet's algorithm.

Let us first formally state our optimization problem. The input is a single graph G and the output is an LC-equivalent graph H with a minimal number of edges. Mathematically it is framed as follows:

$$\begin{aligned} \min_{H} \quad & \sum_{i>j}^{n} A_H[i,j], \\ \text{s.t.} \quad & G \equiv_{\mathrm{LC}} H. \end{aligned}$$

We now utilize the formulation of LC-equivalence found in Eqs. (A7) and (A5). Moreover, since the matrices have binary entries we put that as constraints. This leads to the following equivalent formulation,

$$\begin{aligned}
\min \quad & \sum_{i>j}^{n} A_H[i,j], \qquad \text{(B1)}\\
\text{s.t.} \quad & A_G P A_H + A_G Q + A_H R + S = 0 \mod 2,\\
& A_H \in \{0,1\}^{n\times n},\\
& A_H[i,j] = A_H[j,i],\\
& A_H[i,i] = 0,\\
& P[i,i]S[i,i] + R[i,i]Q[i,i] = 1,\\
& P[i,j], Q[i,j], R[i,j], S[i,j] = 0, i \neq j,\\
& P[i,j], Q[i,j], R[i,j], S[i,j] \in \{0,1\},
\end{aligned}$$

where we write the (i,j)th element of a matrix M as $M[i,j]$. While this is an integer program, it is not an integer *linear* program due to the products of variables as well as the modular constraint. To turn this into an ILP we first remove the modulo 2 constraint by introducing a matrix $B \in \mathbb{Z}^{n\times n}$ of integer variables and demanding that

$$A_G P A_H + A_G Q + A_H R + S = 2B\ .$$

The optimization problem can thus be written as:

$$\begin{aligned}
\min \quad & \sum_{i>j}^{n} A_H[i,j], \qquad \text{(B2)}\\
\text{s.t.} \quad & A_G[i,a]P[a,b]A_H[b,j] + A_G[i,a]Q[a,j]+\\
& \quad A_H[i,a]R[a,j] + S[i,j] = 2B[i,j],\\
& P[i,i]S[i,i] + R[i,i]Q[i,i] = 1,\\
& A_H[i,j] = A_H[j,i],\\
& A_H[i,i] = 0,\\
& A_H[i,j] \in \{0,1\},\\
& P[i,j], Q[i,j], R[i,j], S[i,j] = 0, i \neq j,\\
& P[i,j], Q[i,j], R[i,j], S[i,j] \geq 0.
\end{aligned}$$

We then turn the quadratic constraints (i.e. terms corresponding to $A_G P A_H$, $A_H R$, PS and RQ) into linear constraints by introducing extra variables. For each term, we first replace each product xz of binary variables x and z with a single binary variable y. We then add the following constraints $(1)\, y \leq z$, $(2)\, y \leq x$, $(3)\, y \geq x + z - 1$. One thing to keep in mind is that A_G is a constant, so quadratic terms with A_G do not need to be linearized. We replace each product of the form $P[a,b]A_H[b,j]$ and $A_H[i,a]R[a,j]$ with new corresponding variables Z^P_{abj} and Z^R_{aij}, respectively. For terms of the form PS and RQ we introduce new variables Z^{PS} and Z^{RQ}, respectively. Applying these changes to Eq. (B2) we obtain our `ILP`:

$$\begin{aligned}
\min \quad & \sum_{i>j}^{n} A_H[i,j], \qquad \text{(B3)}\\
\text{s.t.} \quad & A_G[i,a]Z^P_{abj} + A_G[i,a]Q[a,j] + Z^R_{aij} + S[i,j] = 2B[i,j],\\
& Z^P_{abj} \leq A_H[b,j],\\
& Z^P_{abj} \leq P[a,b],\\
& Z^P_{abj} \geq A_H[b,j] + P[a,b] - 1,\\
& Z^R_{aij} \leq A_H[i,a],\\
& Z^R_{aij} \leq R[a,j],\\
& Z^R_{aij} \geq A_H[i,a] + R[a,j] - 1,\\
& Z^{PS}[i,i] \leq S[i,i],\\
& Z^{PS}[i,i] \leq P[i,i],\\
& Z^{PS}[i,i] \geq S[i,i] + P[i,i] - 1,\\
& Z^{RQ}[i,i] \leq R[i,i],\\
& Z^{RQ}[i,i] \leq Q[i,i],\\
& Z^{RQ}[i,i] \geq R[i,i] + Q[i,i] - 1,\\
& A_H[i,j] = A_H[j,i],\\
& A_H[i,i] = 0,\\
& A_H[i,j] \in \{0,1\},\\
& P[i,j], Q[i,j], R[i,j], S[i,j] = 0, i \neq j,\\
& P[i,j], Q[i,j], R[i,j], S[i,j] \in \{0,1\}.
\end{aligned}$$

Using the ILP, we can thus find LC-equivalent graphs to a given graph that have minimum number of edges. We note here that better way of solving the edge-minimization problem using ILP might exist that could allow us to go beyond graph states with 16 qubits.

Appendix C: Choice of neighboring states in `SA`

In our implementation of SA, the state s_k at the k'th iteration is given by a graph. A natural choice for the set $S(s_k)$ of neighboring states is the set of graphs that differ from s_k by a single local complementation. This set could have a maximum size equal to the number of vertices (n) in the graph. However, sometimes there are vertices where local complementation does not have any effect. Moreover, there could also be vertices that lead to isomorphic graphs after applying local complementation, as one could observe in the work of Adcock et al. [34]. If the neighboring states are chosen at random, these vertices could affect the performance of the SA algorithm. Therefore, we use a graph metric called the local clustering coefficient [39] to guide the SA algorithm to decide where local complementation should be applied, thereby improving the efficiency of SA algorithm.

The local clustering coefficient $C_G(v)$ of a vertex of a graph (which will always be clear from the context) provides information on the connectedness of its neighborhood. Specifically, it is the ratio between the number of edges connecting

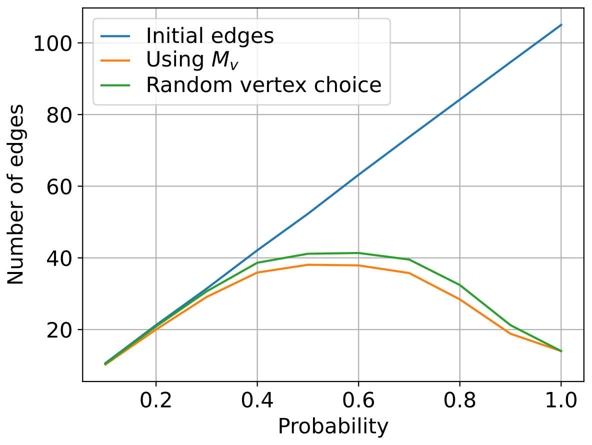


FIG. 5: Here we plot the average edges in the initial graphs (blue line) and the edges obtained from SA using metric M_v (orange) and random vertex choice (green). In the random vertex choice method, a potential solution s_{pot} is found by applying local complementation to a vertex chosen uniformly at random. Initial edges are the edges in the input ER graph. For this run, $k_{\max}$ and $T(1)$ were both set to 100.

the neighbors of the given vertex and the total number of possible edges. This metric has a maximum value of 1 (when the neighborhood is fully connected) and minimum value of zero (when the neighbors share no other edges). That is,

$$C_G(v) = \frac{\text{Number of edges between neighbors}}{\text{Possible number connections between neighbors}}. \tag{C1}$$

A local complementation (L_v) on a vertex v affects its local clustering coefficient $C_G(v)$ in the following way,

$$C_{L_v(G)}(v) = 1 - C_G(v). \tag{C2}$$

Therefore, applying a local complementation to a vertex v with $C_G(v) \geq \frac{1}{2}$ (i.e. a more connected neighborhood) changes the local clustering coefficient to be smaller than $\frac{1}{2}$ (i.e. a less connected neighborhood). Thus, applying local complementations to vertices with a high value of $C_G(v)$ reduces the local edge density. Moreover, to bias vertices that have a larger number of neighbors, we multiply $C_G(v)$ with the degree of the vertex, leading to the heuristic quantity $M_v \equiv C_G(v)D_v$. Based on this heuristic, we decide on the set of states $S(s_k)$ from which the SA algorithm could choose the potential solution s_{pot}. Thus, we bias SA into choosing vertices that have a higher value of M_v.

The algorithm finds the set $S(s_k)$ at the k'th iteration based on the values of heuristic M_v, k and $k_{\max}$. We first construct a set with values of M_v of each vertex in the state s_k. We then find a subset of this set with unique values of M_v. Then, we sample a value from the set of these unique values based on a cutoff. The cutoff is given by $c = k/k_{\max} \times l$, where l is the number of elements in the set of unique values. We randomly select a value from the set of unique values that is greater than the value corresponding to the cutoff; let us call the selected value M_u. Since M_u is in the set of unique values, multiple vertices could have their metrics equal to M_u. Therefore, we find all the vertices that have M_u as their metric value. The graphs obtained by applying local complementation once to these vertices then constitute the set $S(s_k)$. We then sample a state from this set uniformly at random to find the next candidate state s_{pot}.

We benchmark this procedure of sampling vertices using the metric M_v against a procedure where vertices are chosen uniformly at random. We run simulated annealing for ER graphs $G(n, p)$ with 15 vertices for increasing probability p in Fig. 5. We generate 100 graphs for values of $p = \{0.1, 0.2, 0, 3, \ldots, 1\}$ and we apply the SA procedure to each graph. We find that by using M_v as our guiding metric, SA can find better solutions compared to sampling at random.

Appendix D: Generalized repeater graph states

Here, we discuss our method for creating all-photonic generalized repeater graph states (gRGS). We consider using single photon sources and optical fusions [38] to create the desired graph state. To find the optimal number of resources required for creating gRGS, we use the protocol from Ref. [28]. We combine our algorithms with their work to find an improvement in the number of required fusions and the number of resource states (3-qubit GHZ states) for efficiently creating gRGS.

A gRGS has an n-vertex central graph G with leaves L connected to each vertex, as shown in Fig. 3(b)—we will refer to this graph as G+L. Using the SA algorithm, we can reduce the number of edges in the graph G by finding an approximate MER H. Thus, one way to create a gRGS is the Intermediate-LC protocol where we first create the graph H, followed by the application of LC-gates to convert H to G. The leaves are then fused to the central graph G to create G+L.

The above approach is suboptimal, however. This is because the order in which fusions are performed is restricted to first performing the fusions that generate the graph H, while the fusions that involve attaching the leaves need to happen at the end. Attaching individual leaves at the end, one by one, to the central graph G will lead to an exponentially decaying success probability with the number of leaves due to the probabilistic nature of fusions. In general, probabilistic fusions of two graphs of very different sizes are considered suboptimal.

To find a better fusion order, we propose applying LC-gates corresponding to the H to G conversion to the resource states *before* beginning the fusion process. This leads to the following steps: (1) apply LC-gates, (2) follow fusion order to create H, and (3) fuse leaves L. We observe that fusions commute with each other as they act on different qubits. Therefore, we can combine the construction of H and the addition of leaves L into a single step. This simplifies our protocol into two steps: (1) apply LC-gates to suitable qubits of the resource states, (2) follow the fusion order for creating H+L

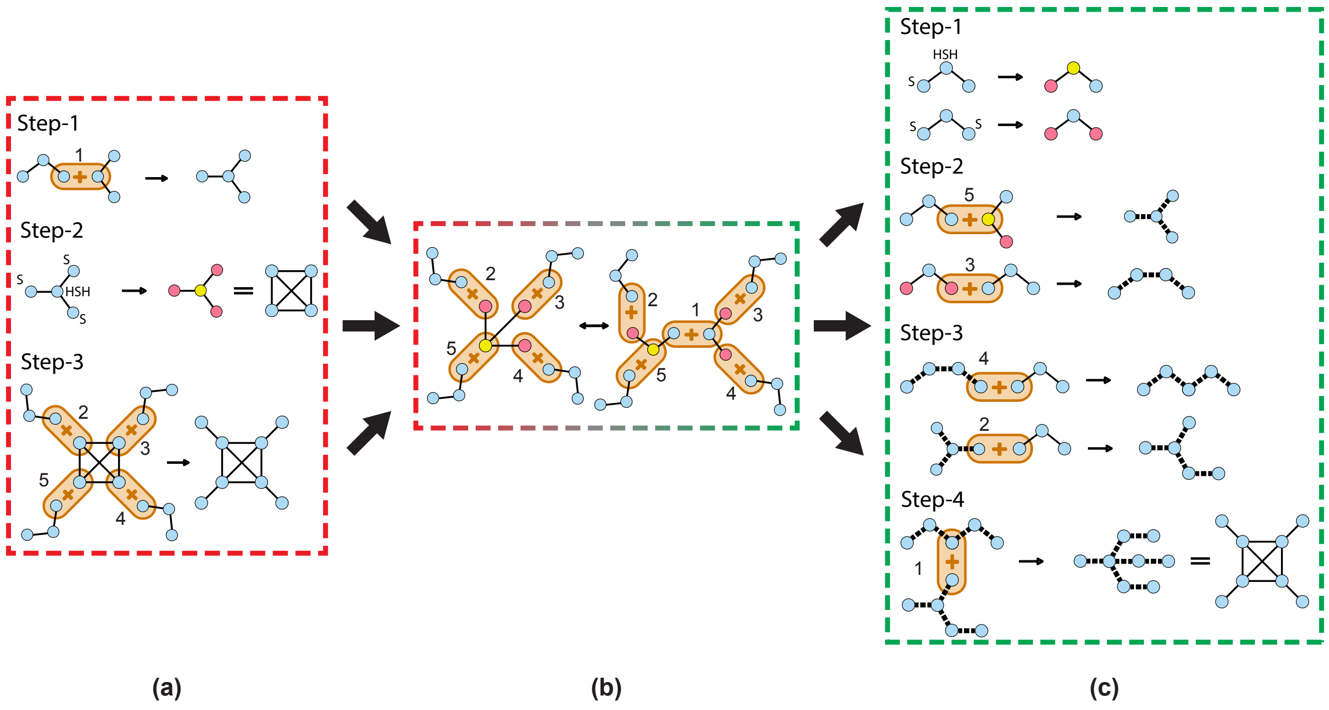


FIG. 6: **(a)** shows the order of fusions for constructing an RGS with 8 qubits following the Intermediate-LC protocol. This order can be used to derive the fusion network shown in **(b)**. The fusion network enables the tracking of the LC-gates relative to fusions for each resource state. Applying LC-gates to resource states in the beginning, as shown in the fusion network, allows the fusions to be reordered flexibly. This reordering can be optimized to reduce the resource requirements. An example of such reordering is the creation of the graph $H{+}L$, as shown in **(c)**.

using these modified resource states. The final state from this fusion order is $G{+}L$ due to the LC-gates applied at the beginning.

Commuting the LC-gates to the beginning of the protocol can be understood by stepping away from the graph state picture and considering the circuit of the protocol. Consider the steps in construction of $G{+}L$ following the Intermediate-LC protocol: (1) construct the graph H, (2) apply LC-gates to convert H to G, and (3) fuse leaves L to the graph G. We make the following observations from the circuit picture: (1) the target qubits for LC-gates have not been involved in any fusions until that point in time (otherwise they would have been measured out). As a result, the LC-gates can be pushed to the beginning of the protocol as they commute with the fusions implemented to construct H. (2) Pauli corrections may be required on qubits adjacent to those involved in fusions. If an LC-gate acts on such a qubit, the corrections can be modified accordingly. This is possible since the Clifford group normalizes the Pauli group, which allows us to commute the LC-gates through the Pauli corrections. Notably, Ref. [28] also applies LC-gates corresponding to local complementations. These gates must also be accounted for when commuting LC-gates to the beginning.

In Fig. 6, we demonstrate the construction of an RGS by commuting LC-gates to the beginning. We first find the suboptimal fusion order for creating $G{+}L$ using Intermediate-LC, shown in Fig. 6(a). We then construct the *fusion network* (similar to Ref. [28]) to determine the location of each LC-gate. A fusion network depicts the order in which each resource state has to be fused. We show the fusion network corresponding to Intermediate-LC in Fig. 6(b). The fusion network also details the information about the LC-gates in the context of the required fusions. We use this information to apply the LC-gates to the appropriate qubits of the resource states at the start. This allows us to reorder the fusions to optimize the number of resources.

The LC-gates corresponding to a local complementation operation on a node v correspond to the application of the HSH gate on the qubit corresponding to node v and the S gate to all the qubits that correspond to the neighbors of node v. These gates altogether convert a graph state into another graph state. Applying a subset of these gates to the graph state will result in a state that is not a graph state but is LC-equivalent to a graph state: a *non-graph state*. When we commute LC-gates corresponding to the conversion of graph H to G, we distribute these gates among multiple resource states, as shown in Fig. 6(c). Thus, the resource states become non-graph states (but are still LC-equivalent) after the application of these LC-gates. We then fuse these non-graph states following the fusion order for creating $H{+}L$ to obtain our desired graph state of $G{+}L$. Thus, the final equality in Step-4 of Fig. 6(c) takes a non-graph state (LHS) to a graph state (RHS).

In conclusion, the Commute-LC protocol enables the creation of G+L by following the procedure for constructing H+L. Importantly, applying LC-gates to the resource states converts them into LC-equivalent non-graph states. The locations of the LC-gates can be identified from the fusion network for creating G+L via Intermediate-LC. This approach has two main advantages: (1) it reduces the number of required fusions and resource states by creating H+L, and (2) it enables obtaining the optimal fusion order using `OptGraphState` from Ref. [28].