

Document Version

Final published version

Licence

CC BY

Citation (APA)

Sharma, H., Goodenough, K., Borregaard, J., Rozpędek, F., & Helsen, J. (2026). Minimising the number of edges in LC-equivalent graph states. *QUANTUM*, 10, 1-21. <https://doi.org/10.22331/q-2026-02-09-2001>

Important note

To cite this publication, please use the final published version (if applicable). Please check the document version above.

Copyright

In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership. Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights. We will remove access to the work immediately and investigate your claim.

Minimising the number of edges in LC-equivalent graph states

Hemant Sharma^{1,2}, Kenneth Goodenough³, Johannes Borregaard⁴, Filip Rozpędek³, and Jonas Helsen²

¹QuTech, Delft University of Technology, 2628 CJ, Delft, The Netherlands

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

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

⁴Department of Physics, Harvard University, Cambridge, Massachusetts 02138, USA

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-minimisation 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 minimising the resources required to create a graph state. We leverage Bouchet’s algebraic formulation of LC-equivalence to encode the edge-minimisation problem as an integer linear program (EDM-ILP). We further propose a simulated annealing (EDM-SA) approach guided by the local clustering coefficient for edge minimisation. We identify new MERs for graph states with up to 16 qubits by combining EDM-SA and EDM-ILP. We extend the ILP to weighted-edge minimisation, where each edge has an associated weight, and prove that this problem is NP-complete. Finally, we employ our tools to minimise the resources required to create all-photonic generalised repeater graph states using fusion operations.

1 Introduction

Graph states are a powerful yet tractable class of multipartite entangled states that have been

Hemant Sharma: h.sharma-1@tudelft.nl

extensively studied in quantum science. Beyond their theoretical significance [1, 2], they support a wide range of applications, including measurement-based quantum computation [3, 4, 5, 6, 7, 8], all-photonic quantum repeaters [9, 10, 11, 12, 13, 14], loss-tolerant error-correcting codes [15, 16, 17], and quantum metrology [18]. Significant efforts have been directed toward both their experimental realisation [19, 20, 21, 22, 23, 24, 25, 26, 27, 28] and theoretical optimisation, encompassing photonic graph state generation [29, 30, 31, 32, 33], general graph state construction [34], and state distribution [35].

A graph state $|G\rangle$ is represented by a graph G with vertices that correspond to qubits initialised 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. [36] 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 [37]. MERs have been studied as a means to reduce the physical resources needed for graph state preparation [34]. However, identify-

ing MERs is challenging due to the convoluted structure of LC-orbits, as shown in Ref [37]. 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 [34, 37].

In this work, we introduce three algorithms to address the edge-minimisation problem: given a graph G , find an LC-equivalent graph H with the minimum number of edges. Our first algorithm, EDM-SA, uses simulated annealing [38] to obtain approximate solutions for graphs with up to 100 qubits. The second algorithm, EDM-ILP, is an integer linear program based on Bouchet’s algebraic characterisation of LC-equivalence [39]. While EDM-ILP guarantees globally optimal solutions, its runtime scales exponentially in the worst case. Combining both approaches, our third method, EDM-SAILP, uses EDM-SA to precondition the input to EDM-ILP, yielding a significant constant-factor speed-up. This hybrid approach enables us to compute exact MERs for graphs with up to 16 qubits. Moreover, we extend EDM-ILP to tackle the weighted-edge minimisation problem: finding LC-equivalent graphs that minimise the total edge weight. We prove this problem is NP-complete via a reduction from the vertex-minor problem [40].

To demonstrate the practical utility of our approach, we use EDM-SA to optimise the resources required for the creation of all-photonic *generalised* repeater graph states (gRGS) [14]. These states can be utilised for sharing multiple units of entanglement in a quantum network. Moreover, they can be constructed using single-photon sources and fusion operations [41]. However, the probabilistic nature of fusion and photon loss leads to a high number of required single photons and fusion operations, with the fusion order critically affecting generation efficiency. To reduce resource requirements, we identify a sub-graph 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 sub-graph, allowing for an optimised fusion order. Using the tools from Ref. [31], 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 2, we provide a description of EDM-SA and EDM-ILP algorithms we use to find MERs. In Section 3, we study the performance of our algorithms. Next, in Section 4, we demonstrate the applications of our framework for edge minimisation. We conclude with an outlook for future work in Section 5.

2 Algorithms for edge minimisation

2.1 Simulated annealing

Here we discuss a heuristic approach for edge minimisation based on simulated annealing (SA) [38]. Due to its flexibility, simulated annealing has been widely applied in various domains, including the optimisation of resource states in measurement-based quantum computation [6, 8], and the distribution of graph states in networks [35].

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 ‘neighbouring’ 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 neighbouring 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 neighbourhood of v , as shown in Fig. 1. We construct the set $S(s_k)$ based on a metric called the local clustering coefficient [42], 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 tempera-

tures. 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).

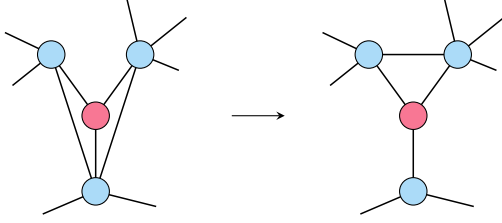


Figure 1: Local complementation on the red vertex removes (or adds) an edge between the neighbours of the vertex if they were connected (or disconnected) prior to its application

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 [43]: at the k 'th iteration, the temperature is given by $T(k) = T(1)/\log_2(k+1)$, for a given initial temperature $T(1)$. Logarithmic cooling schedule is slower compared to other cooling schedules. This allows EDM-SA algorithm to explore the states for longer time before converging to a solution. In principle, the initial temperature and k_{\max} 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.

2.2 Edge minimisation using Bouchet's algorithm

We leverage Bouchet's LC-equivalence between graph states to find the exact solutions to the edge-minimisation problem. By definition, it 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 [39] 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 [44, 45] and Appendix A for further information) to find the following equation describing LC-equivalence of graphs G and H ,

$$A_G P A_H + A_G Q + A_H R + S = 0 \pmod{2}, \quad (1)$$

$$p_i s_i + r_i q_i = 1. \quad (2)$$

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

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

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 linearise 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 $P A_H$ and $A_H R$ in Eq. (4) and terms $P S$ and $R Q$ 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 EDM-ILP algorithm to Appendix B. We note that Eq. (1) form n^2 equations in terms of $4n$ variables, but with the non-linear constraint in Eq. (2). Bouchet showed in [39] that one only needs to check a small number of solutions, to certify whether there exists a solution that also satisfies Eq. (2). In particular, if U forms any basis of the vector space \mathcal{V} of solutions to Eq. (1), then there exists a solution in \mathcal{V} that satisfies the constraint in (4) if and only if there exists an element in the set $\{u + u' \mid u, u' \in U\}$ that satisfies the constraint. Note that this latter set has size at most linear in n , while \mathcal{V} can in principle be of size exponential in n . Unfortunately, we were not able to exploit this structure for the ILP, which

has the potential to reduce the number of constraints and thus the runtime significantly. One obstacle encountered is the fact that bases are not naturally expressed in terms of ILPs, making it hard to exploit the above structure. On the other hand, given access to U , it is possible to express the set $\{u + u' \mid u, u' \in U\}$ naturally in terms of an ILP.

The runtime of the EDM-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 EDM-SA to find the approximate MER of a given graph before inputting it into EDM-ILP. This preprocessing step reduces the number of constraints in the corresponding EDM-ILP thus improving the runtime. We refer to this combined algorithm as EDM-SAILP. We use the MOSEK optimiser API for Python [46] to run EDM-ILP and EDM-SAILP. Since EDM-ILP is an exact optimisation, the results obtained are global minima, i.e. MERs of their corresponding LC-orbit. In Section 3, we benchmark the performance of our algorithms for different graph models.

2.3 Weighted-edge minimisation

Our formulation of Bouchet’s algorithm as an optimisation problem can be easily extended to include weighted-edge minimisation of LC-equivalent graph states. The optimal solution for our case is thus the one that minimises the sum of these weights of LC-equivalent graph states. The optimisation 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 possible pair 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. [35]. We allow the weights $W[i, j]$ to be real numbers, since an ILP only requires the variables to be integers, not the cost function. Note that the weights $W[i, j]$ are separate from the weights of *weighted graph states*, studied in e.g. [47, 48], which are not the subject of our work.

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-minimisation problem. The vertex-minor (or qubit-minor) problem plays an important role in quantum information theory [49, 50]. 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 $|G\rangle$ [49, 50]. Accordingly, the graph H is called a vertex-minor of graph G [51]. We consider the labelled 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 labelled [50] and unlabelled case [40].

The vertex-minor problem can be formulated using the weighted-edge minimisation problem as follows. Assume that the graph $H = (V', E')$ has no isolated vertices¹. 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 \bar{G} such that $\bar{G}[V'] = H$, 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 [40], it follows immediately that the decision problem for weighted-edge minimisation is NP-complete as well. We note that this still leaves open whether the minimisation problem remains NP-complete when restricting to uniform/non-negative weights.

¹For the case of H containing isolated vertices, assign a weight of 0 to any of the edges incident to an isolated vertex.

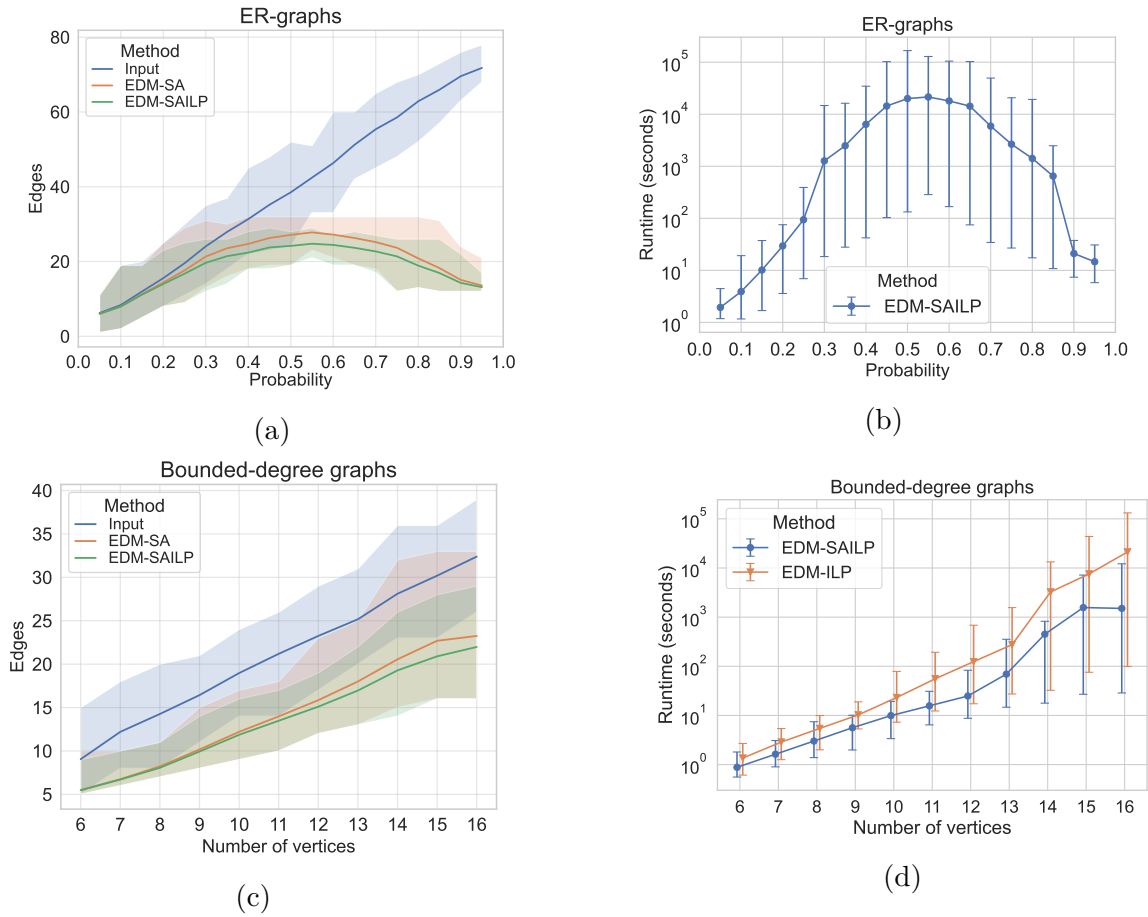


Figure 2: In **(a)** and **(c)**, we compare the number of edges in the input graphs and the outputs from EDM-SA and EDM-SAILP for Erdős-Rényi graphs and bounded-degree graphs respectively. The solid lines denote the average value, and the shaded bands indicate the maximum and minimum. **(b)** shows the runtime of the EDM-SAILP algorithm as a function of edge inclusion probability for ER graphs with 13 vertices. **(d)** compares the runtime between EDM-ILP and EDM-SAILP 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 EDM-SAILP achieves a constant-factor reduction in runtime compared to EDM-ILP on average. This improvement can be attributed to the EDM-SA preprocessing step, although the runtime remains exponential in the size of the input graph. The average runtime of EDM-SAILP for graphs with 16 vertices is smaller than that for graphs with 15 vertices, which we attribute to a better EDM-SA performance.

3 Edge minimisation of LC-equivalent graph states

In this section, we first characterise the runtime of EDM-SA. We then empirically evaluate: (1) the runtime of EDM-ILP and EDM-SAILP, and (2) the reduction in edge count achieved by EDM-SA and EDM-SAILP. 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 [52, 53]. 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 characterise the runtime of the EDM-SA algorithm by running it for different numbers of maximum iterations. The initial temperature is

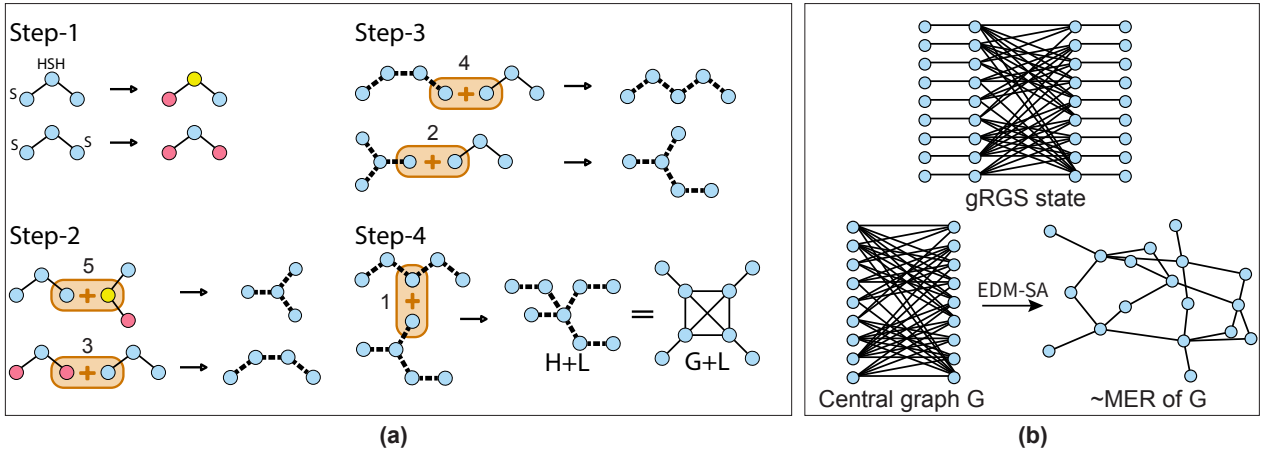


Figure 3: **(a)** demonstrates the fusion order for creating an RGS with eight qubits, obtained using `OptGraphState` from Ref. [31]. The elliptical boxes show the required fusions, labelled by numbers next to them. These numbers allow the tracking of fusions after LC-gates are commuted to the beginning. 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.

set to $T(1) = 50$ for a set of 100 Erdős-Rényi graphs with 100 vertices and 2971.2 edges on average. For $k_{\max} = 50$, EDM-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. This suggests that larger values of k_{\max} might allow for better performance of EDM-SA. However, 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 EDM-SA and EDM-SAILP to minimise the edges in each graph generated for a p value. From Fig. 2(a), we find that the number of edges obtained from EDM-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 EDM-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 EDM-SAILP 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 EDM-ILP algorithm for ER graphs in Fig. 2(b) since the number of edges grows to a large number (~ 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 [54, 55]. In our simulations, we fixed the value of d_{lim} to 5 and generated graphs with n ranging from 6 to 16. We generated 100 graphs for each vertex number; however for $n = 6$ only 63 such graphs were generated.

In Fig. 2(c), we benchmark the edge minimisation performance of EDM-SA against the MERs obtained from EDM-SAILP for bounded-degree graphs. The number of edges obtained from EDM-SA is on average 1.04 times larger than the MERs. We observe that the performance of EDM-SA declines slightly as the number of vertices increases, likely due to the increased sparsity of the input graphs. In addition to edge count comparison, we compare the runtime of EDM-

ILP and EDM-SAILP in Fig. 2(d). To find the dependence of runtime of EDM-SAILP, we calculate the correlation coefficient between the logarithm of its runtime and the number of edges and vertices input to the EDM-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 EDM-SA as a pre-processing step to EDM-ILP reduces the runtime of the EDM-SAILP algorithm by a factor of two compared to EDM-ILP alone for bounded-degree graphs. Thus, EDM-SAILP is a faster algorithm than EDM-ILP but it still has an exponential runtime dependence on the number of edges and vertices in the input graphs.

4 Creation of generalised 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) [9] 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 [29, 30, 56, 57] but often comes with a large overhead in the number of required single photon sources. A more general class of all-photonic states called *generalised* repeater graph states (gRGS) was introduced in Ref. [14], which offers a better scaling between photon overhead and the number of shared ebits. Here, we apply our MER algorithms to further minimise 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 [41]. 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 [58].

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. [31] 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. [31] addresses these challenges and reduces the resource overhead for photonic graph state creation by optimising (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 generalised repeater graph states to optimise 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 EDM-SA algorithm. Naively, one could fuse resource states to create $G+L$ directly. However, a more resource-efficient method for creating 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.

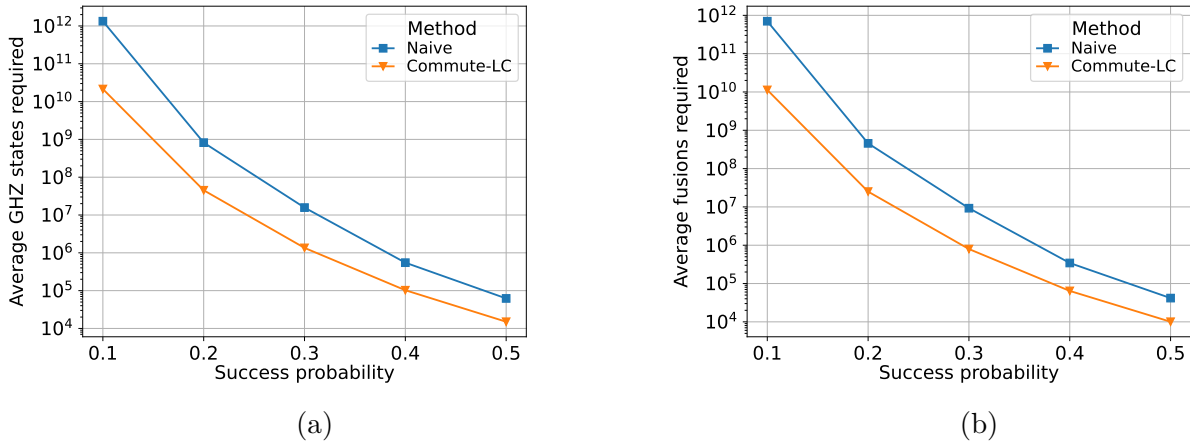


Figure 4: **(a)** and **(b)** show the required number of resource states and fusions, respectively, for creating a single instance of $G+L$ gRGS with 64 edges and 36 vertices (see Fig. 3(b)). We compare the naive approach of direct gRGS construction (blue curve) with the Commute-LC protocol (orange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

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. [31] also applies LC-gates to their qubits during their optimisation. However, these LC-gates only convert a fully-connected sub-graph (clique) into a star. In contrast, by applying EDM-SA to the central graph G in Protocols 1 and 2, we generalise the use of LC-gates to simplify the fusion process not just for cliques but for arbitrary densely connected sub-graphs.

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 ², 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

²Similar to Ref. [14]

this graph with 23 edges using the EDM-SA algorithm (with $k_{\max} = 10000$ 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.

5 Outlook

We have proposed three algorithms to address the edge-minimisation problem. First, a simulated annealing (EDM-SA) approach efficiently finds approximate MERs for graphs with up to 100 qubits. Second, we introduce an integer linear programming (EDM-ILP) formulation that yields exact solutions, albeit with potentially exponential runtime. Third, we combine both methods into a hybrid EDM-SAILP 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 minimisation and prove that this variant is NP-complete. Importantly, we show that the scalable EDM-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. Moreover, the performance of EDM-SA could still be improved by tailoring the initial temperature and maximum iterations for

specific problem instances.

We have applied our EDM-SA algorithm to minimise the required resources for all-photonic generalised 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 EDM-SA can be combined with existing methods from Ref. [31] to reduce the required number of single-photon sources and fusion operations by more than an order of magnitude for realistic lossy implementations.

Our results open 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. [34]. 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 [59, 60, 61]. 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 [62]. Graph states under local unitary equivalence are significantly less well-understood, but this topic has seen recent investigations [63, 64, 65], closely relating this equivalence to the Clifford hierarchy. One natural follow-up question is whether edge minimisation of graphs under local unitaries can similarly be phrased as an ILP (using, for example, the algorithm from [65]), and, if so, to what extent this coarser equivalence can reduce the edge count.

Acknowledgements

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. FR acknowledges support from NSF (NSF grant No. ERC-1941583). 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

The package is publicly available on PyPI and can be installed with `pip install graphstate-opt`. Moreover, a release of our code is also available at [66] and the accompanying data at [67], with the expectation that it will serve as a valuable resource for the quantum information community in addressing practical optimisation 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.

References

- [1] M. Hein, J. Eisert, and H. J. Briegel. “Multiparty entanglement in graph states”. *Phys. Rev. A* **69**, 062311 (2004).
- [2] Marc Hein, Wolfgang Dür, Jens Eisert, Robert Raussendorf, Maarten Van den Nest, and H-J Briegel. “Entanglement in graph states and its applications”. In *Quantum computers, algorithms and chaos*. Pages 115–218. IOS Press (2006).
- [3] Robert Raussendorf and Hans J. Briegel. “A one-way quantum computer”. *Phys. Rev. Lett.* **86**, 5188–5191 (2001).
- [4] Robert Raussendorf, Daniel E. Browne, and Hans J. Briegel. “Measurement-based quantum computation on cluster states”. *Phys. Rev. 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**, 19–26 (2009).

- [6] Thierry N Kaldenbach and Matthias Heller. “Mapping quantum circuits to shallow-depth measurement patterns based on graph states”. *Quantum Science and Technology* **10**, 015010 (2024).
- [7] Madhav Krishnan Vijayan, Alexandru Paler, Jason Gavriel, Casey R Myers, Peter P Rohde, and Simon J Devitt. “Compilation of algorithm-specific graph states for quantum circuits”. *Quantum Science and Technology* **9**, 025005 (2024).
- [8] Thierry N. Kaldenbach, Isaac D. Smith, Hendrik Poulsen Nautrup, Matthias Heller, and Hans J. Briegel. “Efficient preparation of resource states for hamiltonian simulation and universal quantum computation” (2025). [arXiv:2509.05404](https://arxiv.org/abs/2509.05404).
- [9] Koji Azuma, Kiyoshi Tamaki, and Hoi-Kwong Lo. “All-photonic quantum repeaters”. *Nature Communications* **6** (2015).
- [10] Mihir Pant, Hari Krovi, Dirk Englund, and Saikat Guha. “Rate-distance tradeoff and resource costs for all-optical quantum repeaters”. *Phys. Rev. A* **95**, 012304 (2017).
- [11] Filip Rozpędek, Kaushik P. Seshadreesan, Paul Polakos, Liang Jiang, and Saikat Guha. “All-photonic Gottesman-Kitaev-Preskill-qubit repeater using analog-information-assisted multiplexed entanglement ranking”. *Phys. Rev. Res.* **5**, 043056 (2023).
- [12] Eneet Kaur, Ashlesha Patil, and Saikat Guha. “Resource-efficient loss-aware photonic-graph-state preparation using atomic emitters”. *Phys. Rev. A* **112**, 062608 (2025).
- [13] Ashlesha Patil and Saikat Guha. “An improved design for all-photonic quantum repeaters” (2024). [arXiv:2405.11768](https://arxiv.org/abs/2405.11768).
- [14] Bikun Li, Kenneth Goodenough, Filip Rozpędek, and Liang Jiang. “Generalized quantum repeater graph states”. *Phys. Rev. Lett.* **134**, 190801 (2025).
- [15] Johannes Borregaard, Hannes Pichler, Tim Schröder, Mikhail D. Lukin, Peter Lodahl, and Anders S. Sørensen. “One-way quantum repeater based on near-deterministic photon-emitter interfaces”. *Phys. Rev. X* **10**, 021071 (2020).
- [16] Thomas J. Bell, Love A. Pettersson, and Stefano Paesani. “Optimizing graph codes for measurement-based loss tolerance”. *PRX Quantum* **4**, 020328 (2023).
- [17] Ashlesha Patil and Saikat Guha. “Tree cluster state generation using percolation”. In *Optica Quantum 2.0 Conference and Exhibition. Page QTh4A.3. QUANTUM*. Optica Publishing Group (2023).
- [18] Nathan Shettell and Damian Markham. “Graph states as a resource for quantum metrology”. *Phys. Rev. Lett.* **124**, 110502 (2020).
- [19] 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**, 434–437 (2016).
- [20] Mikkel V. Larsen, Xueshi Guo, Casper R. Breum, Jonas S. Neergaard-Nielsen, and Ulrik L. Andersen. “Deterministic generation of a two-dimensional cluster state”. *Science* **366**, 369–372 (2019).
- [21] Warit Asavanant, Yu Shiozawa, Shota Yokoyama, Baramée Charoensombutamon, Hiroki Emura, Rafael N. Alexander, Shuntaro Takeda, Jun-ichi Yoshikawa, Nicolas C. Menicucci, Hidehiro Yonezawa, and Akira Furusawa. “Generation of time-domain-multiplexed two-dimensional cluster state”. *Science* **366**, 373–376 (2019).
- [22] D. Istrati, Y. Pilnyak, J. C. Loredó, C. Antón, N. Somaschi, P. Hilaire, H. Ollivier, M. Esmann, L. Cohen, L. Vidro, C. Millet, A. Lemaitre, I. Sagnes, A. Harouri, L. Lanco, P. Senellart, and H. S. Eisenberg. “Sequential generation of linear cluster states from a single photon emitter”. *Nature Communications* **11** (2020).
- [23] Philip Thomas, Leonardo Ruscio, Olivier Morin, and Gerhard Rempe. “Efficient generation of entangled multiphoton graph states from a single atom”. *Nature* **608**, 677–681 (2022).

- [24] Chan Roh, Geunhee Gwak, Young-Do Yoon, and Young-Sik Ra. “Generation of three-dimensional cluster entangled state”. *Nature Photonics* **19**, 526–532 (2025).
- [25] Sirui Cao, Bujiao Wu, Fusheng Chen, et al. “Generation of genuine entanglement up to 51 superconducting qubits”. *Nature* **619**, 738–742 (2023).
- [26] James O’Sullivan, Kevin Reuer, et al. “Deterministic generation of two-dimensional multi-photon cluster states”. *Nature Communications* **16** (2025).
- [27] Vinicius S Ferreira, Gihwan Kim, Andreas Butler, Hannes Pichler, and Oskar Painter. “Deterministic generation of multidimensional photonic cluster states with a single quantum emitter”. *Nature Physics* **20**, 865–870 (2024).
- [28] Philip Thomas, Leonardo Ruscio, Olivier Morin, and Gerhard Rempe. “Fusion of deterministically generated photonic graph states”. *Nature* **629**, 567–572 (2024).
- [29] Bikun Li, Sophia E. Economou, and Edwin Barnes. “Photonic resource state generation from a minimal number of quantum emitters”. *npj Quantum Information* **8** (2022).
- [30] Evangelia Takou, Edwin Barnes, and Sophia E Economou. “Optimization complexity and resource minimization of emitter-based photonic graph state generation protocols”. *npj Quantum Information* **11**, 108 (2025).
- [31] Seok-Hyung Lee and Hyunseok Jeong. “Graph-theoretical optimization of fusion-based graph state generation”. *Quantum* **7**, 1212 (2023).
- [32] Yingheng Li, Yue Dai, Aditya Pawar, Rongchao Dong, Jun Yang, Youtao Zhang, and Xulong Tang. “Reinforcement learning-guided graph state generation in photonic quantum computers”. In Proceedings of the 52nd Annual International Symposium on Computer Architecture. Page 1598–1612. SIGARCH ’25. ACM (2025).
- [33] Daniel Bhatti and Kenneth Goode-nough. “Distributing graph states with a photon-weaving quantum server” (2025). [arXiv:2504.07410](https://arxiv.org/abs/2504.07410).
- [34] Adán Cabello, Lars Eirik Danielsen, Antonio J. López-Tarrida, and José R. Portillo. “Optimal preparation of graph states”. *Phys. Rev. A* **83**, 042314 (2011).
- [35] Tingxiang Ji, Jianqing Liu, and Zheshen Zhang. “Distributing arbitrary quantum graph states by graph transformation” (2025). [arXiv:2404.05537](https://arxiv.org/abs/2404.05537).
- [36] Maarten Van den Nest, Jeroen Dehaene, and Bart De Moor. “Graphical description of the action of local clifford transformations on graph states”. *Phys. Rev. A* **69**, 022316 (2004).
- [37] Jeremy C. Adcock, Sam Morley-Short, Axel Dahlberg, and Joshua W. Silverstone. “Mapping graph state orbits under local complementation”. *Quantum* **4**, 305 (2020).
- [38] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. “Optimization by simulated annealing”. *Science* **220**, 671–680 (1983).
- [39] André Bouchet. “An efficient algorithm to recognize locally equivalent graphs”. *Combinatorica* **11**, 315–329 (1991).
- [40] Axel Dahlberg, Jonas Helsen, and Stephanie Wehner. “The complexity of the vertex-minor problem”. *Information Processing Letters* **175**, 106222 (2022).
- [41] Daniel E. Browne and Terry Rudolph. “Resource-efficient linear optical quantum computation”. *Phys. Rev. Lett.* **95**, 010501 (2005).
- [42] Duncan J Watts and Steven H Strogatz. “Collective dynamics of ‘small-world’ networks”. *Nature* **393**, 440–442 (1998).
- [43] Bruce Hajek. “Cooling schedules for optimal annealing”. Page 147–150. Springer New York. (1987).
- [44] Jeroen Dehaene, Maarten Van den Nest, Bart De Moor, and Frank Verstraete. “Local permutations of products of bell states and entanglement distillation”. *Phys. Rev. A* **67**, 022310 (2003).

- [45] Jeroen Dehaene and Bart De Moor. “Clifford group, stabilizer states, and linear and quadratic operations over $\text{gf}(2)$ ”. *Phys. Rev. A* **68**, 042318 (2003).
- [46] MOSEK ApS. “Mosek optimizer api for python 10.2.1”. (2024). url: <https://docs.mosek.com/latest/pythonapi/index.html>.
- [47] Simon Anders, Hans J Briegel, and Wolfgang Dür. “A variational method based on weighted graph states”. *New Journal of Physics* **9**, 361–361 (2007).
- [48] L Hartmann, J Calsamiglia, W Dür, and H J Briegel. “Weighted graph states and applications to spin chains, lattices and gases”. *Journal of Physics B: Atomic, Molecular and Optical Physics* **40**, S1 (2007).
- [49] Axel Dahlberg and Stephanie Wehner. “Transforming graph states using single-qubit operations”. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **376**, 20170325 (2018).
- [50] Axel Dahlberg, Jonas Helsen, and Stephanie Wehner. “How to transform graph states using single-qubit operations: computational complexity and algorithms”. *Quantum Science and Technology* **5**, 045016 (2020).
- [51] Sang-il Oum. “Rank-width and vertex-minors”. *Journal of Combinatorial Theory, Series B* **95**, 79–100 (2005).
- [52] P. Erdős and A. Rényi. “On random graphs. i.”. *Publicationes Mathematicae Debrecen* **6**, 290–297 (2022).
- [53] E. N. Gilbert. “Random graphs”. *The Annals of Mathematical Statistics* **30**, 1141–1144 (1959).
- [54] 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**, 496–506 (1962).
- [55] Václav Havel. “A remark on the existence of finite graphs”. *Časopis pro pěstování matematiky* **080**, 477–480 (1955).
- [56] Donovan Buterakos, Edwin Barnes, and Sophia E. Economou. “Deterministic generation of all-photon quantum repeaters from solid-state emitters”. *Phys. Rev. X* **7**, 041023 (2017).
- [57] Yuan Zhan, Paul Hilaire, Edwin Barnes, Sophia E. Economou, and Shuo Sun. “Performance analysis of quantum repeaters enabled by deterministically generated photonic graph states”. *Quantum* **7**, 924 (2023).
- [58] Michael Varnava, Daniel E. Browne, and Terry Rudolph. “How good must single photon sources and detectors be for efficient linear optical quantum computation?”. *Phys. Rev. Lett.* **100**, 060502 (2008).
- [59] Soh Kumabe, Ryuhei Mori, and Yusei Yoshimura. “Complexity of graph-state preparation by clifford circuits” (2025). [arXiv:2402.05874](https://arxiv.org/abs/2402.05874).
- [60] Andrew Jena. “Graph-theoretic techniques for optimizing nisq algorithms” (2024).
- [61] James Davies and Andrew Jena. “Preparing graph states forbidding a vertex-minor” (2025). [arXiv:2504.00291](https://arxiv.org/abs/2504.00291).
- [62] Z.-F. Ji, J.-X. Chen, Z.-H. Wei, and M.-S. Ying. “The lu-lc conjecture is false”. *Quantum Information and Computation* **10**, 97–108 (2010).
- [63] Adam Burchardt, Jarn de Jong, and Lina Vandr . “Algorithm to verify local equivalence of stabilizer states” (2025). [arXiv:2410.03961](https://arxiv.org/abs/2410.03961).
- [64] Nathan Claudet and Simon Perdrix. “Local Equivalence of Stabilizer States: A Graphical Characterisation”. In 42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025). *Volume 327 of Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:18. Dagstuhl, Germany (2025). Schloss Dagstuhl – Leibniz-Zentrum f r Informatik.
- [65] Nathan Claudet and Simon Perdrix. “Deciding local unitary equivalence of graph states in quasi-polynomial time”. In LIPIcs, Volume 334, ICALP 2025. *Volume 334, pages 59:1–59:20*. Schloss Dagstuhl – Leibniz-Zentrum f r Informatik (2025).

- [66] Hemant Sharma. “Release 1.0.1: graph_state_optimization”. <https://doi.org/10.5281/zenodo.15534878> (2025).
- [67] Hemant Sharma. “Data accompanying graph_state_optimization. this includes the sampled graphs and mers corresponding to those graphs.”. <https://doi.org/10.5281/zenodo.15534839> (2025).
- [68] Maarten Van den Nest, Jeroen Dehaene, and Bart De Moor. “Efficient algorithm to recognize the local clifford equivalence of graph states”. *Phys. Rev. A* **70**, 034302 (2004).
- [69] Sobhan Ghanbari, Jie Lin, Benjamin MacLellan, Luc Robichaud, Piotr Roztock, and Hoi-Kwong Lo. “Optimization of deterministic photonic-graph-state generation via local operations”. *Phys. Rev. A* **110**, 052605 (2024).

A LC-equivalence algorithm for graph states

We will now briefly recap Bouchet's algorithm, using the stabilizer formalism (as was done in [68]). In particular, we will focus on the commonly used symplectic framework of the stabilizer formalism [44, 68]. 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 $\phi(CsC^\dagger) = M\phi(s)$, for every Pauli string s . The matrix M being symplectic means that

$$M^T \Omega M = \Omega, \quad \text{where } \Omega = \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}, \quad (6)$$

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

$$\begin{bmatrix} A_G \\ I \end{bmatrix}, \quad \begin{bmatrix} A_H \\ I \end{bmatrix}, \quad (7)$$

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

$$M \begin{bmatrix} A_H \\ I \end{bmatrix} W = \begin{bmatrix} A_G \\ I \end{bmatrix} \quad (8)$$

$$M = \begin{bmatrix} P & Q \\ R & S \end{bmatrix}, \quad W \in \text{GL}_{2n}(\mathbb{F}_2). \quad (9)$$

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

$$p_i s_i + r_i q_i = 1, \quad (10)$$

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 [68], 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

$$\begin{bmatrix} A_G & I \end{bmatrix} \Omega M \begin{bmatrix} A_H \\ I \end{bmatrix} = 0. \quad (11)$$

Expanding Eq. (11) yields:

$$A_G P A_H + A_G Q + A_H R + S = 0. \quad (12)$$

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. (10).

Eq. (12) is a set of n^2 equations in $4n$ variables (p_i, q_i, r_i, s_i) with a constraint (Eq. (10)) that is non-linear, so in principle not easily solvable. However, through an ingenious argument, Bouchet [39] proved that only a small number of solutions to Eq. 12 have to be checked to verify if there exists one that also satisfies Eq. (10), making the total runtime polynomial [39]. However for edge minimisation, we have unfortunately not been able to exploit this extra structure with our ILP (see the main text). As such, we just use Eqs. 12 and 10 to construct our ILPs.

B Derivation of EDM-ILP

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

Let us first formally state our optimisation 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:

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

We now utilise the formulation of LC-equivalence found in Eqs. (12) and (10). Moreover, since the matrices have binary entries we put that as constraints. This leads to the following equivalent formulation,

$$\begin{aligned} \min \quad & \sum_{i>j}^n A_H[i, j], & (13) \\ \text{s.t.} \quad & A_G P A_H + A_G Q + A_H R + S = 0 \pmod{2}, \\ & A_H \in \{0, 1\}^{n \times n}, \\ & A_H[i, j] = A_H[j, i], \\ & A_H[i, i] = 0, \\ & P[i, i] S[i, i] + R[i, i] Q[i, i] = 1, \\ & P[i, j], Q[i, j], R[i, j], S[i, j] = 0, i \neq j, \\ & P[i, j], Q[i, j], R[i, j], S[i, j] \in \{0, 1\}, \end{aligned}$$

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 optimisation problem can thus be written as:

$$\begin{aligned}
\min \quad & \sum_{i>j}^n A_H[i, j], \\
\text{s.t.} \quad & A_G[i, a]P[a, b]A_H[b, j] + A_G[i, a]Q[a, j] + \\
& A_H[i, a]R[a, j] + S[i, j] = 2B[i, j], \\
& P[i, i]S[i, i] + R[i, i]Q[i, i] = 1, \\
& A_H[i, j] = A_H[j, i], \\
& A_H[i, i] = 0, \\
& A_H[i, j] \in \{0, 1\}, \\
& P[i, j], Q[i, j], R[i, j], S[i, j] = 0, i \neq j, \\
& P[i, j], Q[i, j], R[i, j], S[i, j] \geq 0.
\end{aligned} \tag{14}$$

We then turn the quadratic constraints (i.e. terms corresponding to A_GPA_H , A_HR , 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 linearised. 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_{abj}^P and Z_{aij}^R , respectively. For terms of the form PS and RQ we introduce new variables Z^{PS} and Z^{RQ} , respectively. Applying these changes to Eq. (14) we obtain our EDM-ILP:

$$\begin{aligned}
\min \quad & \sum_{i>j}^n A_H[i, j], \\
\text{s.t.} \quad & A_G[i, a]Z_{abj}^P + A_G[i, a]Q[a, j] + \\
& Z_{aij}^R + S[i, j] = 2B[i, j], \\
& Z_{abj}^P \leq A_H[b, j], \\
& Z_{abj}^P \leq P[a, b], \\
& Z_{abj}^P \geq A_H[b, j] + P[a, b] - 1, \\
& Z_{aij}^R \leq A_H[i, a], \\
& Z_{aij}^R \leq R[a, j], \\
& Z_{aij}^R \geq A_H[i, a] + R[a, j] - 1, \\
& Z^{PS}[i, i] \leq S[i, i], \\
& Z^{PS}[i, i] \leq P[i, i], \\
& Z^{PS}[i, i] \geq S[i, i] + P[i, i] - 1, \\
& Z^{RQ}[i, i] \leq R[i, i], \\
& Z^{RQ}[i, i] \leq Q[i, i], \\
& Z^{RQ}[i, i] \geq R[i, i] + Q[i, i] - 1, \\
& A_H[i, j] = A_H[j, i], \\
& A_H[i, i] = 0, \\
& A_H[i, j] \in \{0, 1\}, \\
& P[i, j], Q[i, j], R[i, j], S[i, j] = 0, i \neq j, \\
& P[i, j], Q[i, j], R[i, j], S[i, j] \in \{0, 1\}.
\end{aligned} \tag{15}$$

Using the ILP, we can thus find LC-equivalent graphs to a given graph that have minimum number of

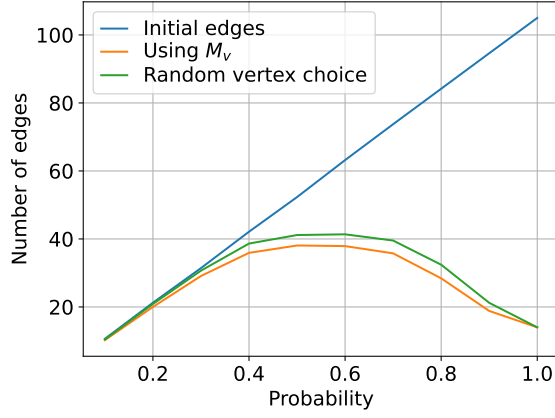


Figure 5: Here we plot the average edges in the initial graphs (blue line) and the edges obtained from EDM-SA using metric M_v (orange) and random vertex choice (green). In the random vertex choice method, a potential solution s_{pot} is found by applying local complementation to a vertex chosen uniformly 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.

edges. We note here that better ways of solving the edge-minimisation problem using ILP might exist that could allow us to go beyond graph states with 16 qubits.

C Choice of neighbouring states in EDM-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 neighbouring 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. [37]. If the neighbouring 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 [42] 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 neighbourhood. Specifically, it is the ratio between the number of edges connecting the neighbours of the given vertex and the total number of possible edges. This metric has a maximum value of 1 (when the neighbourhood is fully connected) and minimum value of zero (when the neighbours share no other edges). That is,

$$C_G(v) = \frac{\text{Number of edges between neighbours}}{\text{Possible connections between neighbours}}. \quad (16)$$

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

$$C_{L_v(G)}(v) = 1 - C_G(v). \quad (17)$$

Therefore, applying a local complementation to a vertex v with $C_G(v) \geq \frac{1}{2}$ (i.e. a more connected neighbourhood) changes the local clustering coefficient to be smaller than $\frac{1}{2}$ (i.e. a less connected neighbourhood). 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 neighbours, we multiply $C_G(v)$ with the degree of the vertex, leading to the heuristic quantity $M_v \equiv C_G(v)D_v$, similar to [69]. 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 .

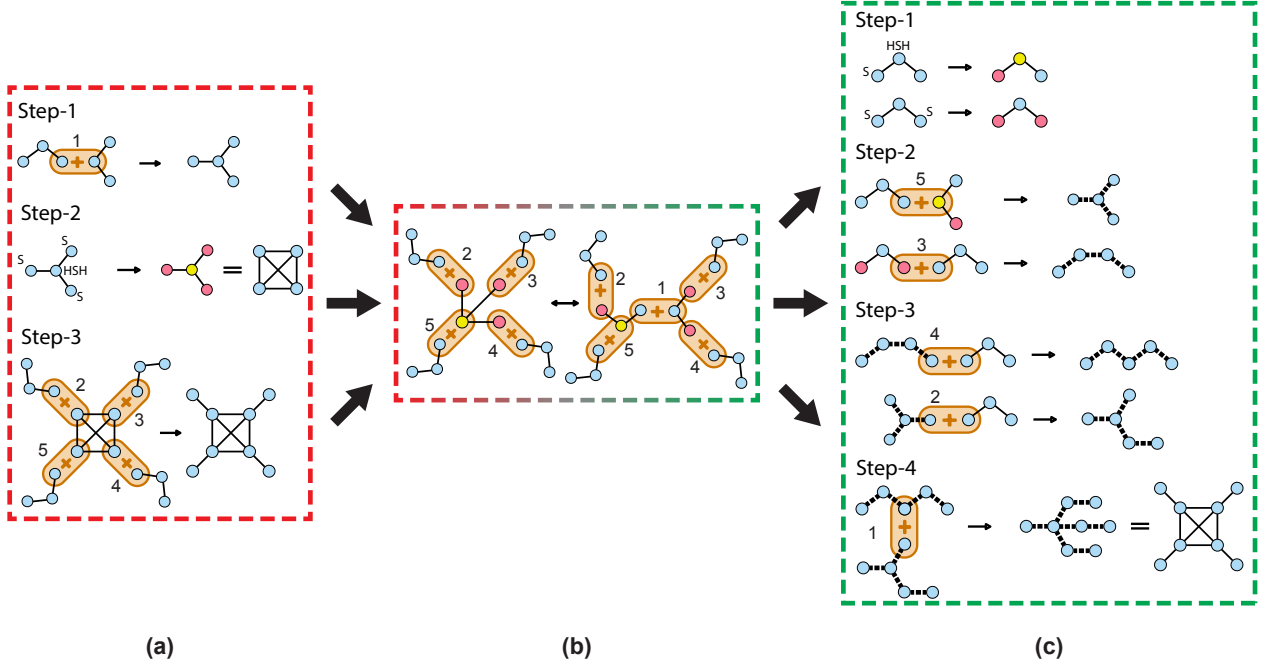


Figure 6: **(a)** shows the order of fusions for constructing an RGS with 8 qubits following the Intermediate-LC protocol. This order can be used to derive the fusion network shown in **(b)**. The fusion network enables the tracking of the LC-gates relative to fusions for each resource state. To track each fusion operation, we label each fusion with a number next to it. These numbers allow us to keep track of the fusion operations after we commute the LC-operations through the fusions. 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 optimised to reduce the resource requirements. An example of such reordering is the creation of the graph $H+L$, as shown in **(c)**.

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 $p = \{0.1, 0.2, 0.3, \dots, 1\}$ 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.

D Generalised repeater graph states

Here, we discuss our method for creating all-photonic generalised repeater graph states (gRGS). We consider using single photon sources and optical fusions [41] to create the desired graph state. To find the optimal number of resources required for creating gRGS, we use the protocol from Ref. [31]. 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 EDM-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 $H+L$ 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 normalises the Pauli group, which allows us to commute the LC-gates through the Pauli corrections. Notably, Ref. [31] 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. [31]) 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 optimise 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 neighbours 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. [31].

E Example of MERs

In this section we illustrate the difference between the edge minimisation performance of EDM-SA and EDM-SAILP.

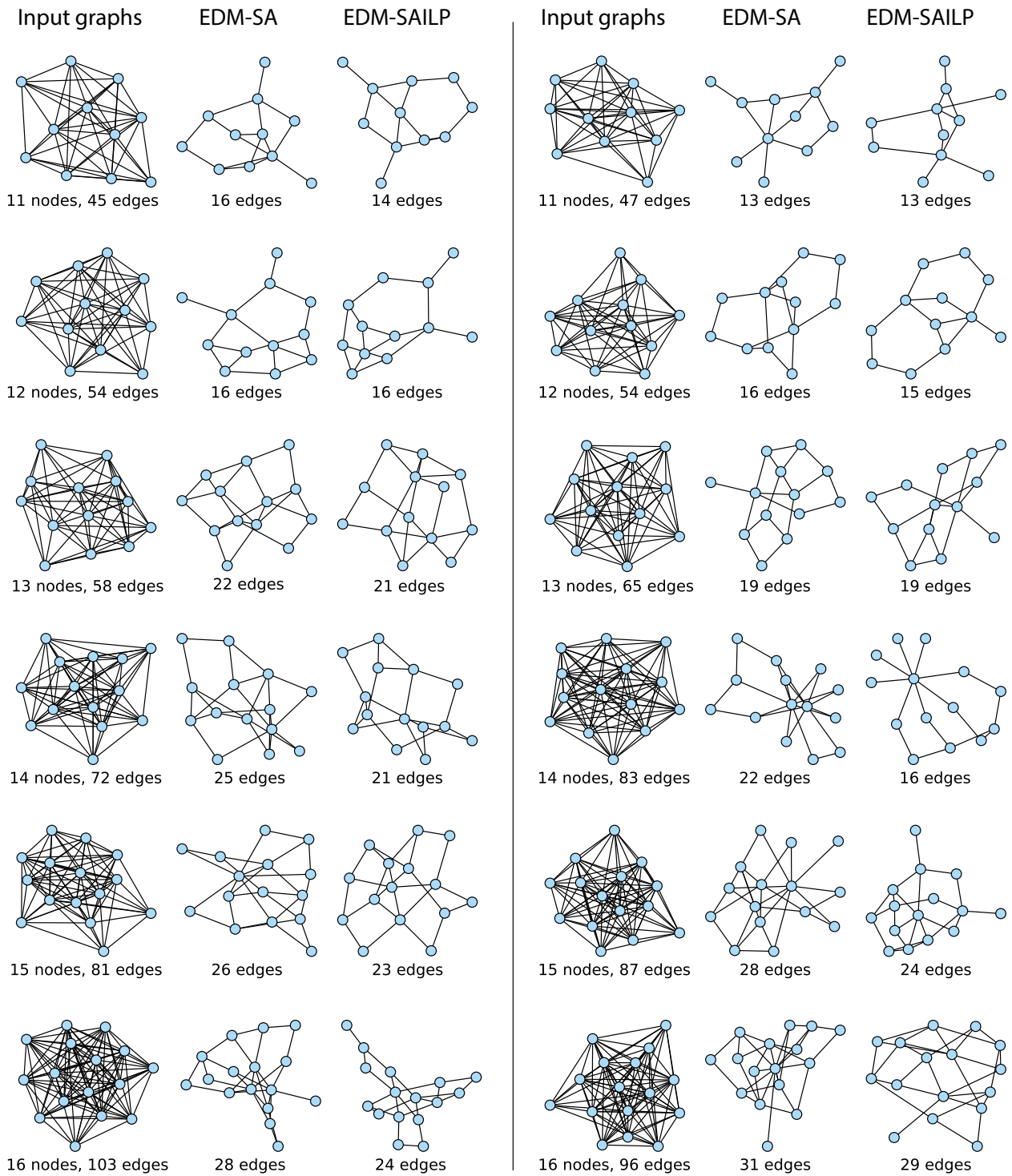


Figure 7: We sample ER graphs $G(n, p)$ with increasing n from 11 to 16 and $p = 0.8$. We run EDM-SA and EDM-SAILP on these graphs, with k_{\max} and $T(1)$ set to 100. We illustrate a set of three graphs, namely: (1) input graphs, (2) approximate MERs obtained from EDM-SA and (3) the exact MERs obtained from EDM-SAILP. Each row has two such sets with the same number of vertices, and the number of vertices increase from 11 to 16 over all rows. We can observe that both the algorithms provide significant reductions in the number of edges, this is partly be due to the input graph being dense. We also observe that in some cases the number of edges in the approximate MERs obtained from EDM-SA are a little higher than the MERs.