Graph-theoretical optimization of fusion-based graph state generation

Seok-Hyung Lee[1,2] and Hyunseok Jeong[1]

[1]Department of Physics and Astronomy, Seoul National University, Seoul 08826, Republic of Korea

[2]Centre for Engineered Quantum Systems, School of Physics, University of Sydney, Sydney, NSW 2006, Australia

Graph states are versatile resources for various quantum information processing tasks, including measurement-based quantum computing and quantum repeaters. Although the type-II fusion gate enables all-optical generation of graph states by combining small graph states, its non-deterministic nature hinders the efficient generation of large graph states. In this work, we present a graph-theoretical strategy to effectively optimize fusion-based generation of any given graph state, along with a Python package *OptGraphState*. Our strategy comprises three stages: simplifying the target graph state, building a fusion network, and determining the order of fusions. Utilizing this proposed method, we evaluate the resource overheads of random graphs and various well-known graphs. Additionally, we investigate the success probability of graph state generation given a restricted number of available resource states. We expect that our strategy and software will assist researchers in developing and assessing experimentally viable schemes that use photonic graph states.

Graph states represent a family of multi-qubit states where qubits are entangled following a specific structure determined by an associated graph. Owing to their highly entangled nature [1], graph states find applications in various quantum information processing domains, such as measurement-based quantum computing (MBQC) [2, 3, 4, 5], fusion-based quantum computing (FBQC) [6], quantum error correction [7, 8], quantum secret sharing [9, 10], quantum repeaters [11, 12, 13, 14], and quantum metrology [15]. Photonic qubit-based graph states are particularly crucial in these applications, not only because photons are predominantly used in quantum communication but also because MBQC can circumvent the need for in-line near-deterministic multi-qubit gates that are challenging in photonic systems [16].

Seok-Hyung Lee: seokhyung.lee@sydney.edu.au

Hyunseok Jeong: h.jeong37@gmail.com

All-optical methods for constructing photonic graph states are commonly processed by merging multiple smaller graph states into a larger one using *fusion operations of types I and/or II* [17]. The failures of these operations are heralded, presenting a significant advantage [18] over alternative methods such as the post-selected controlled-Z (CZ) gate [19, 20] and the post-selected fusion gate [17]. Among the two fusion types, we focus exclusively on type II. This is because, assuming photodetectors with negligible dark counts, a type-I fusion could potentially transform a photon loss into an undetectable computational error [21], whereas any photon loss occurring before or during a type-II fusion can be identified.

The non-deterministic nature of fusions is a crucial consideration when investigating quantum tasks using photonic graph states. When employing dual-rail-encoded qubits (such as polarization qubits) and restricting the setup to linear-optical devices and photodetectors, the success probability of a type-II fusion is limited to 50% without ancillary resources [22]. Higher success probabilities can be achieved by utilizing ancillary photons [23, 24], encoded qubits [25, 26], redundant graph structures [27, 28, 21], or continuous-variable techniques [29, 30, 31, 32, 33, 34]. Through these methods, fault-tolerant linear-optical MBQC is theoretically possible. For instance, our recent research verified that high photon loss thresholds of around 8% under a uniform photon loss model can be attained by employing parity-encoded multiphoton qubits [35].

Despite these advancements, resource overhead remains a significant challenge for generating large-scale graph states. Specifically, the number of required basic resource states (such as three-photon Greenberger–Horne–Zeilinger states) or optical elements like photodetectors increases exponentially as the number of fusions grows. Consequently, it is essential to carefully design a procedure for generating a desired graph state from basic resource states to minimize resource overhead as much as possible. While several prior studies [36, 37] have addressed this issue, they only considered specific graph families and type-I fusion. In our previous work [35], we proposed a partial solution for general graphs and type-II fusion using a fine-tuning strategy; however, there is still considerable scope for improvement.

In this work, we introduce a graph-theoretical strategy to effectively identify a resource-efficient method for fusion-based generation of any given graph state, building upon and generalizing the strategies presented in Ref. [35]. A single trial of our strategy comprises three main stages: (i) simplifying the graph state through the process of *unraveling*, (ii) constructing a *fusion network* (a graph that dictates the required fusions between basic resource states), and (iii) determining the order of fusions. A sufficient number of trials are repeated with randomness and the one with the smallest resource overhead is selected as the outcome of the strategy. Although our approach does not guarantee the most optimal method, we provide evidence of its power and generality, making it suitable for studying various tasks involving graph states. Our strategy is implemented in an open-source Python package, *OptGraphState*, which is publicly available on Github: `https://github.com/seokhyung-lee/OptGraphState`.

This paper is structured as follows: In Sec. 1, we review the definitions and basic properties of graph states and type-II fusion. In Sec. 2, we describe our optimization strategy step by step with examples. In Sec. 3, we compute the resource overheads of various graphs using our strategy and numerically verify its effectiveness by comparing it with alternative strategies that lacks certain stages of the original strategy. We additionally discuss the success probability of generating a graph state given a restricted number of available basic resource states. We conclude with final remarks in Sec. 4.

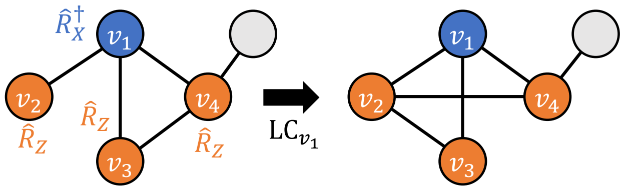

Figure 1: **Example of a local complementation (LC) and the corresponding single-qubit Clifford operations.** $\hat{R}_X$ and $\hat{R}_Z$ indicate a $\pi/2$ rotation around the x- and z-axis, respectively, in the Bloch sphere; namely, $\hat{R}_P := \exp\left[-i(\pi/4)\hat{P}\right]$ for $\hat{P} \in \left\{\hat{X}, \hat{Z}\right\}$. For the presented five-qubit graph state, applying $\hat{R}_X^\dagger$ to vertex v_1 and $\hat{R}_Z$ to each of its neighbors is equivalent to transform the graph by an LC with respect to v_1.

1 Preliminaries

1.1 Graph states and their equivalence relation

For a given graph $G = (V,\ E)$ with a vertex set V and an edge set E, a graph state $|G\rangle_V$ on qubits placed at the vertices is defined as

$$|G\rangle_V := \prod_{\{v_1,v_2\}\in E} \hat{U}_{\mathrm{CZ}}(v_1,v_2) \bigotimes_{v\in V} |+\rangle_v \,,$$

where $\hat{U}_{\mathrm{CZ}}(v_1,v_2)$ is the controlled-Z (CZ) gate between the qubits at v_1 and v_2 and $|+\rangle_v$ is the state $|0\rangle + |1\rangle$ on v. (We omit normalization coefficients throughout the paper unless necessary.) The graph state $|G\rangle_V$ has the stabilizer group generated by

$$\left\{ \hat{S}_v := \hat{X}_v \prod_{u\in\mathrm{adj}(v)} \hat{Z}_u \;\middle|\; v \in V \right\},$$

where $\hat{X}_v$ and $\hat{Z}_v$ are respectively the Pauli-X and Z operators on v and $\mathrm{adj}(v)$ is the set of the adjacent vertices of v. Namely, $\hat{S}_v |G\rangle_V = |G\rangle_V$ for every $v \in V$.

An important problem regarding graph states is whether two different graph states are equivalent under a unitary operation, especially under a sequence of single-qubit Clifford operations. For a graph $G = (V, E)$ and a vertex $v \in V$, we define a Clifford operator

$$\hat{U}_{\mathrm{LC}}(v) := e^{i\frac{\pi}{4}\hat{X}_v} \prod_{u\in\mathrm{adj}(v)} e^{-i\frac{\pi}{4}\hat{Z}_u}. \qquad (1)$$

A *local complementation* LC_v with respect to a vertex $v \in V$ is defined as a graph operation

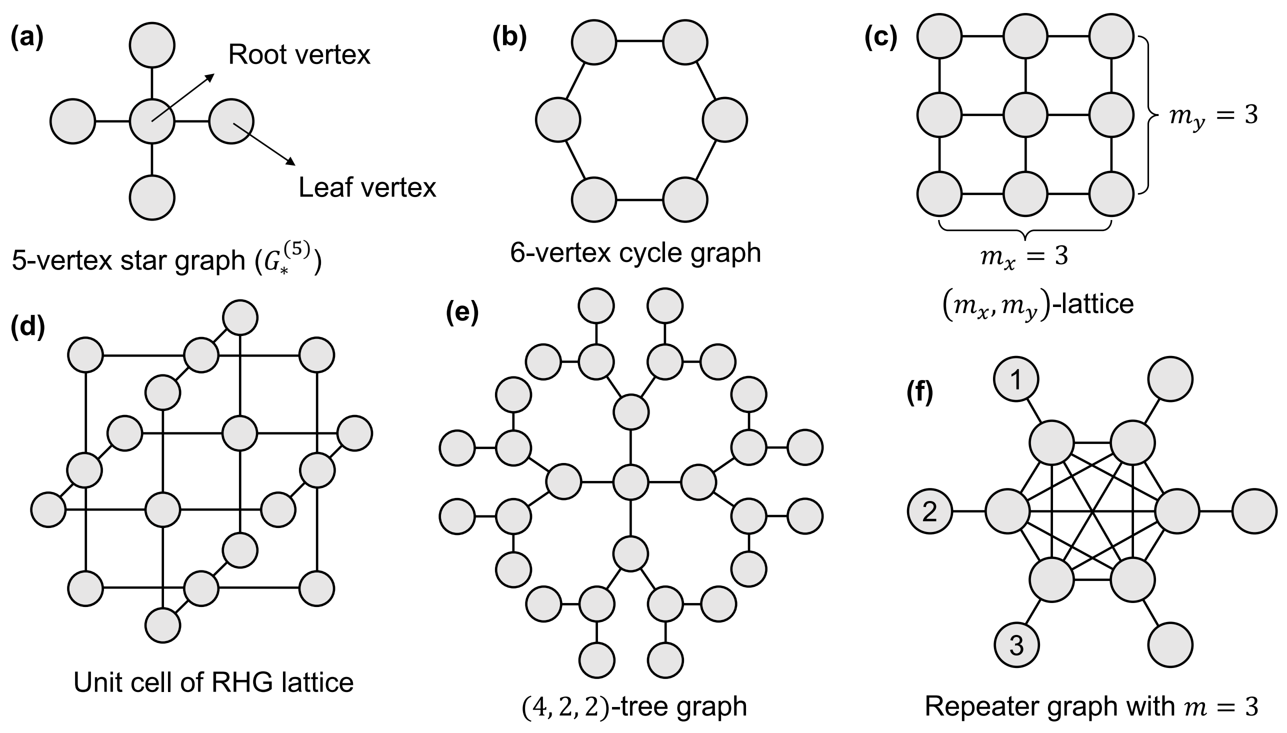

Figure 2: **Examples of various well-known families of graphs.** See Sec. 1.1 for their descriptions and usages.

that, for every pair of adjacent vertices of v, connect them if they are disconnected and disconnect them if they are connected. As proved in Ref. [38] and visualized in Fig. 1, $\hat{U}_{\mathrm{LC}}(v)$ transforms G by a local complementation; namely,

$$\hat{U}_{\mathrm{LC}}(v) |G\rangle = |\mathrm{LC}_v(G)\rangle .$$

Furthermore, it is known that two graph states are equivalent under a sequence of single-qubit Clifford operations if and only if one of their corresponding graphs can be obtained by applying a sequence of local complementations on the other [38].

The followings describe several well-known families of graph states visualized in Fig. 2:

Star graph. The m-vertex star graph $G_*^{(m)}$ is a graph where one of the vertices (say, v_{root}) is connected with all the other vertices that are not connected with each other; see Fig. 2(a) for an example. The vertex v_{root} is called the *root* vertex of $G_*^{(m)}$ and the other vertices are called its *leaf* vertices. Note that the graph state $\left|G_*^{(m)}\right\rangle$ is equivalent to the m-qubit Greenberger–Horne–Zeilinger (GHZ) state $|\mathrm{GHZ}_m\rangle := |0\rangle^{\otimes m} + |1\rangle^{\otimes m}$ and the graph state of the m-vertex complete graph $G_{\mathrm{cmpl}}^{(m)}$ (where all the vertices are connected) under single-qubit Clifford gates; namely,

$$|\mathrm{GHZ}_m\rangle = \left(\prod_{v \in V_{\mathrm{leaf}}} \hat{H}_v\right) \left|G_*^{(m)}\right\rangle ,$$
$$\left|G_{\mathrm{cmpl}}^{(m)}\right\rangle = \hat{U}_{\mathrm{LC}}(v_{\mathrm{root}}) \left|G_*^{(m)}\right\rangle ,$$

where V_{leaf} is the set of the leaf vertices of $G_*^{(m)}$ and $\hat{H}_v$ is the Hadamard gate on the qubit at v. Star graph states are often used as basic resource states of photonic MBQC [28, 21, 31, 39, 35] and FBQC [6].

Cycle graph. A cycle graph consists of vertices connected in a closed chain. In particular, the graph state for the six-vertex cycle graph, which is shown in Fig. 2(b), is used as a basic resource state of FBQC [6].

Lattice graph. The (m_x, m_y)-lattice graph for integers $m_x, m_y \geq 1$ has a two-dimensional (2D) square lattice structure where the vertices are repeated m_x (m_y) times along the x-axis (y-axis). See Fig. 2(b) for an example. Lattice graph states are particularly useful for 2D MBQC [2, 3], which is universal but not fault-tolerant [4]. Any single-qubit rotation and the controlled-NOT gate can be implemented by measuring qubits of a lattice graph state in appropriate single-qubit bases.

Raussendorf-Harrington-Goyal (RHG) lattice. The (L_x, L_y, L_z)-RHG lattice graph is composed of unit cells stacked L_x, L_y, and L_z times along the x-, y-, and z-axis, respectively. Each unit cell is cube-shaped and it consists of vertices locating at the faces and edges of the cube, as visualized in Fig. 2(d). RHG lattices are utilized in fault-tolerant three-dimensional (3D) MBQC [4, 5]. Logical qubits encoded in a surface code can be embedded into a lattice and logical operations and measurements can be done only by single-qubit measurements and state injection. A specific operator on each unit cell serves as a parity-check operator, whose measurement outcome is used to detect and locate errors.

Tree graph. A tree graph is defined as a connected acyclic graph. We particularly define the $(b_0, b_1, b_2, \cdots)$-tree graph for positive integers $b_0, b_1, b_2, \cdots$ as a tree graph where a vertex (designated its *root* vertex) has b_0 neighbors called 1st-generation *branches* and each ith-generation branch ($i \geq 1$) has b_i+1 neighbors that are $(i+1)$-generation branches except for one. As an example, see Fig. 2(e) for the $(4, 2, 2)$-tree graph. One important usage of tree graphs is counterfactual error correction; by attaching tree graph states on qubits for 2D MBQC, qubit loss up to 50% can be tolerated [40]. Such a technique also can be employed for 3D MBQC to suppress the effects of failed entangling operations during the construction of an RHG lattice [21].

Repeater graph. The $4m$-vertex repeater graph ($m \geq 1$) consists of $2m$ completely-connected vertices and other $2m$ vertices that are respectively connected with them; see Fig. 2(f) for the case of $m = 3$. A repeater graph state can be used for all-optical quantum repeaters [13], which distribute entanglement over a long distance by recursive entanglement swapping. m determines the number of Bell-state measurements (BSMs) required per single trial of entanglement swapping, which succeeds if any one of these m BSMs succeed.

1.2 Type-II fusion operation

The *type-II fusion operation* (hereafter referred to simply as "fusion") [17] is a two-qubit operation that consists of applying a Hadamard gate

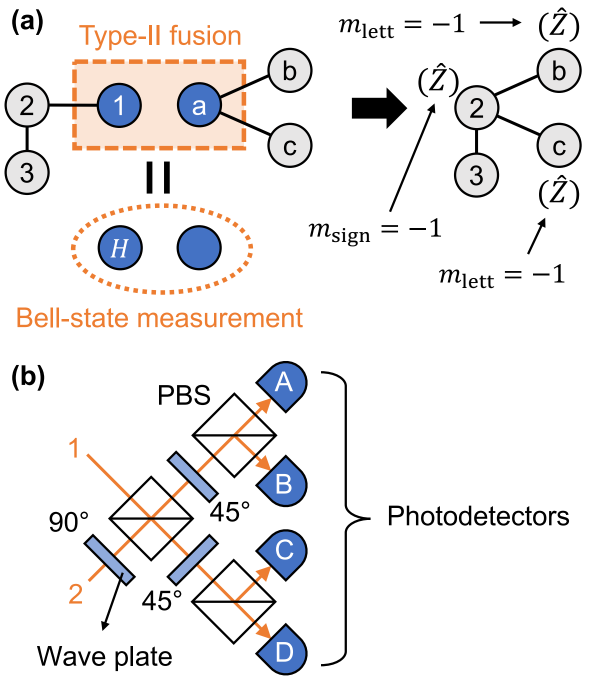

Figure 3: **Type-II fusion operation.** **(a)** Example of merging two graph states by a type-II fusion, which is composed of a Hadamard gate $(\hat{H})$ and a Bell-state measurement (BSM). m_{sign} and m_{lett} respectively denote the sign and letter outcomes of the BSM. If m_{sign} or m_{lett} is -1, several Pauli-Z gates need to be applied on the resulting state to get the graph state. **(b)** BSM scheme for single-photon polarization qubits with polarizing beam splitters (PBSs), $90°$ and $45°$ wave plates, and four (A–D) photodetectors. A PBS transmits (reflects) horizontally-polarized (vertically-polarized) photons. The scheme distinguishes $|\psi_\pm\rangle$: $|\psi_+\rangle$ if both A and C or both B and D detect a single photon respectively, and $|\psi_-\rangle$ if both A and D or both B and C detect a single photon respectively. If otherwise, the scheme fails. Two distinguishable Bell states can be selected by putting or removing wave plates before the first PBS.

$(\hat{H})$ to one of the qubits, followed by a BSM, and finally erasing the qubits. In other words, fusion indicates a destructive measurement of two Pauli operators $\hat{X}\otimes\hat{Z}$ and $\hat{Z}\otimes\hat{X}$ on a pair of qubits. By applying a fusion on an unconnected pair (v_1, v_2) of vertices in a graph state, we can connect (disconnect) every adjacent vertex of v_1 with every adjacent vertex of v_2 up to several Pauli-Z operators if they are unconnected (connected); see the example in Fig. 3(a).

More formally, for two unconnected vertices v_1 and v_2 of a graph G, F_{v_1,v_2} is defined as a graph operation that, for every $u_1 \in \text{adj}(v_1)$ and $u_2 \in \text{adj}(v_2)$, connect (disconnect) u_1 and u_2 if they are unconnected (connected) and delete v_1 and

v_2 from the graph. When v_1 and v_2 undergo a fusion for which the Hadamard gate is applied on v_1, the resulting state is

$$\prod_{u_1 \in \mathrm{adj}\,(v_1)} \hat{Z}_{u_1}^{\frac{1-m_{\mathrm{sign}}}{2}} \prod_{u_2 \in \mathrm{adj}\,(v_2)} \hat{Z}_{u_2}^{\frac{1-m_{\mathrm{lett}}}{2}} \left|F_{v_1,v_2}(G)\right\rangle,$$

where $(m_{\mathrm{sign}}, m_{\mathrm{lett}})$ is the outcome of the BSM. Here, we denote the Bell basis as

$$|\phi_\pm\rangle := |00\rangle \pm |11\rangle\,, \tag{2}$$
$$|\psi_\pm\rangle := |01\rangle \pm |10\rangle\,, \tag{3}$$

and the outcome of a BSM as $(\pm 1, 1)$ if $|\phi_\pm\rangle$ is obtained and $(\pm 1, -1)$ if $|\psi_\pm\rangle$ is obtained. Note that fusion was originally defined as a BSM in Ref. [17]. Nevertheless, we consider its variant (differing only by a Hadamard gate) since it is more suitable to generate arbitrary graph states due to its aforementioned property.

For single-photon polarization qubits with the basis of horizontally and vertically polarized single-photon states ($|\mathrm{H}\rangle$, $|\mathrm{V}\rangle$), the BSM can be done with linear optical devices and photodetectors [41], as visualized in Fig. 3(b). This BSM scheme can distinguish only two among the four Bell states, thus the fusion succeeds with the probability of

$$p_{\mathrm{succ}}(\eta) = \frac{(1-\eta)^2}{2}$$

when each photon suffers loss with probability η and the input state is maximally mixed. See Ref. [35] for a discussion on how failed fusions affect the resulting graph state.

2 Strategy for identifying a method for graph state generation

In this section, we present our main result: a graph-theoretical strategy to effectively identify a resource-efficient method for generating an arbtirary graph state via fusions. Our basic resource state is the three-qubit star graph state

$$\left|G_*^{(3)}\right\rangle = |\text{○-○-○}\rangle := |{+}0{+}\rangle + |{-}1{-}\rangle\,, \tag{4}$$

where $|\pm\rangle := |0\rangle \pm |1\rangle$. Hence, our goal is to find an efficient way to build a desired graph state $|G\rangle$ by performing fusions on multiple $\left|G_*^{(3)}\right\rangle$ states. We take an approach that, whenever a fusion fails, we discard the photons that were entangled with the photons involved in the failed fusion, regenerate the state of the discarded photons with new photons, and retry the fusion. This process does not necessarily have to proceed in order; that is, we may parallelly generate multiple identical states and post-select only the ones with successful fusions.

The resource overhead of a method to generate $|G\rangle$ is quantified by the expected value Q of the number of $\left|G_*^{(3)}\right\rangle$'s required to generate one $|G\rangle$ state. For example, a $\left|G_*^{(4)}\right\rangle$ state can be generated by fusing two $\left|G_*^{(3)}\right\rangle$'s, thus this process has the resource overhead of four if $p_{\mathrm{succ}} = 1/2$. Fusing it again with another $\left|G_*^{(3)}\right\rangle$ state gives the resource overhead of $2 \times (4+1) = 10$. In reality, the success probability may vary for each fusion due to different lengths of delay lines, but we simplify the problems by neglecting this fact in the following investigations. Note that, by definition, having Q basic resource states does not guarantee the successful generation of the graph state. The exact success probability is worth investigating, which will be covered later in Sec. 3.4.

The basic resource state $\left|G_*^{(3)}\right\rangle$ can be generated with a success rate of $1/32$ (in a lossless case) by using linear optical devices, single-photon sources, and photodetectors [42]. Furthermore, its deterministic generation is possible with matter-based methods [43, 44], which is experimentally demonstrated in several recent works [45, 46, 47].

The key concept of our strategy is a *fusion network*, which is a graph where vertices correspond to individual $\left|G_*^{(3)}\right\rangle$ states and edges indicate fusions between the states required to generate $|G\rangle$. Since fusion networks are not unique, selecting an appropriate fusion network is the first challenge. The second challenge is to determine the order of the fusions; although the final state is regardless of the order as long as all the fusions succeed, the non-deterministic nature of fusions makes it severely affect the resource efficiency.

The strategy is summarized as follows:

1. Simplify the graph of the desired graph state by *unraveling* subgraphs of specific types (bipartitely-complete subgraphs and cliques). (Sec. 2.1)

2. Construct a fusion network from the simpli-

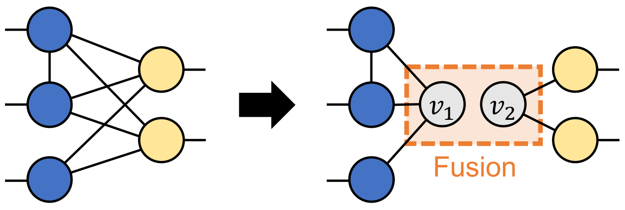

Figure 4: **Example of an unraveling process of a bipartitely-complete subgraph.** The vertices of the two parts are colored in blue and yellow, respectively. The original graph state can be constructed by performing a fusion on v_1 and v_2 of the unraveled graph state.

fied graph by decomposing it into multiple star graphs and replacing each star graph with multiple $\left|G_*^{(3)}\right\rangle$ states. (Sec. 2.2)

3. Determine the fusion order with the *min-weight-maximum-matching-first* method. (Sec. 2.3)

4. Iterate the above steps (which contain randomness) a sufficient number of times and select the best one. (Sec. 2.4)

We cover these steps in the following four subsections one by one.

2.1 Simplification of graph by unraveling

If the graph $G = (V, E)$ of the desired graph state $|G\rangle$ contains specific types of subgraphs, it is posible to generate $|G\rangle$ by applying single-qubit Clifford operations and/or fusions on the graph state of a simplified graph. *Unraveling* means the process to build such a simplified graph $G_{\text{unrv}} = (V', E')$ (referred to as an *unraveled graph*) and specify the information $\left(\hat{U}_{\text{C}}, \mathcal{F}\right)$ necessary to recover $|G\rangle$ from $|G_{\text{unrv}}\rangle$, where $\hat{U}_{\text{C}}$ is the product of single-qubit Clifford operations and $\mathcal{F} \subset V' \times V'$ is the set of pairs of vertices that undergo fusions. We currently have unraveling schemes for two types of subgraphs: bipartitely-complete subgraphs and cliques.

2.1.1 Unraveling bipartitely-complete subgraphs

Definition 2.1 (Bipartitely-complete graph/subgraph). A graph $G = (V, E)$ is an (n, m) *bipartitely-complete graph* for an integer $n, m \geq 2$ if V can be split into two disjoint subsets V_1 and V_2 such that $|V_1| = n$, $|V_2| = m$, and each vertex of V_1 is connected with each vertex of V_2 (namely, $\{v_1, v_2\} \in E$ for any $v_1 \in V_1$ and $v_2 \in V_2$). If a subgraph of a graph is an (n, m) bipartitely-complete graph, it is called an (n, m) *bipartitely-complete subgraph (BCS)*.

Note that a bipartitely-complete graph allows edges between vertices in one part, thus it is different from a complete bipartite graph. The parameter (n, m) of a bipartitely-complete graph may not be uniquely determined.

If G has an (n, m) BCS, its two parts can be disconnected by adding two vertices (v_1, v_2) that are respectively connected with all the vertices in one of the two parts and adding (v_1, v_2) to $\mathcal{F}$, which replace nm edges with $n+m$ edges and one fusion; see Fig. 4 for an example. This process is called unraveling the BCS.

In our strategy, we repeat the cycle of finding non-overlapping BCSs (that do not share any vertices) via Algorithm 1 and unraveling them as above until no new BCSs are found. The time complexity of Algorithm 1 is $\mathcal{O}(|V| d_{\max}^4)$ in the worst case, where $d_{\max}$ is the largest degree[1]. Note that the iterations in Algorithm 1 are done in random orders because the final unraveled graph may vary depending on the orders and we want to suppress any possible bias during iteration (Step 4 of the strategy). All the randomness appearing from now on exist for the same reason.

2.1.2 Unraveling cliques

Definition 2.2 (Clique). A *clique* of a graph G is a subgraph of G where every vertex is fully connected with each other. A clique is *maximal* if it cannot be enlarged by adding a new vertex.

If G contains a clique, it can be simplified by using a local complementation. For a maximal clique of size greater than two, the unraveling process is conducted as follows (see Fig. 5 for an example):

1. Let us define V_{cl} as the set of the vertices in the clique and $V_{\text{no.outer}} \subseteq V_{\text{cl}}$ as the set of the vertices in the clique that are connected only with vertices in the clique.

2. If $V_{\text{no.outer}}$ is not empty, select a vertex v_0 randomly from $V_{\text{no.outer}}$.

[1]The degree of a vertex is the number of its adjacent vertices.

Algorithm 1: Finding non-overlapping bipartitely-complete subgraphs (BCSs)

```
Input: A graph $G = (V, E)$.
Output: A set $\mathcal{S}$ of BCSs of $G$ that do not share any vertices.
1  $\mathcal{S} \leftarrow \emptyset$;
2  $V_{\text{in.bcs}} \leftarrow \emptyset$;  // Vertices included in any BCS found so far
3  $E_{\text{checked}} \leftarrow \emptyset$;  // Edges checked so far
4  foreach $v \in V$ (in a random order) do
5      if $v \notin V_{\text{in.bcs}}$ then
6          $V_{\text{adj.unchecked}} \leftarrow \{v_{\text{adj}} \in \text{adj}(v) \mid \{v, v_{\text{adj}}\} \notin E_{\text{checked}}\}$;
7          foreach unordered pair $\{v_1, v_2\}$ of elements in $V_{\text{adj.unchecked}}$ (in a random order) do
8              if $v_1, v_2 \notin V_{\text{in.bcs}}$ then
9                  $V_1 \leftarrow \text{adj}(v_1) \cap \text{adj}(v_2)$;  // First part of BCS, which includes $v$
10                 $V_2 \leftarrow \bigcap_{u \in V_1} \text{adj}(u)$;  // Second part of BCS, which includes $v_1$ and $v_2$
11                 if $|V_1| > 1$ and $|(V_1 \cup V_2) \cap V_{\text{in.bcs}}| = 0$ then
12                     $\mathcal{S} \leftarrow \mathcal{S} \cup \{(V_1, V_2)\}$;
13                     $V_{\text{in.bcs}} \leftarrow V_{\text{in.bcs}} \cup V_1 \cup V_2$;
14         end
15         foreach $v_{\text{adj}} \in \text{adj}(v) \setminus V_{\text{in.bcs}}$ do
16             $E_{\text{checked}} \leftarrow E_{\text{checked}} \cup \{\{v, v_{\text{adj}}\}\}$;
17         end
18 end
```

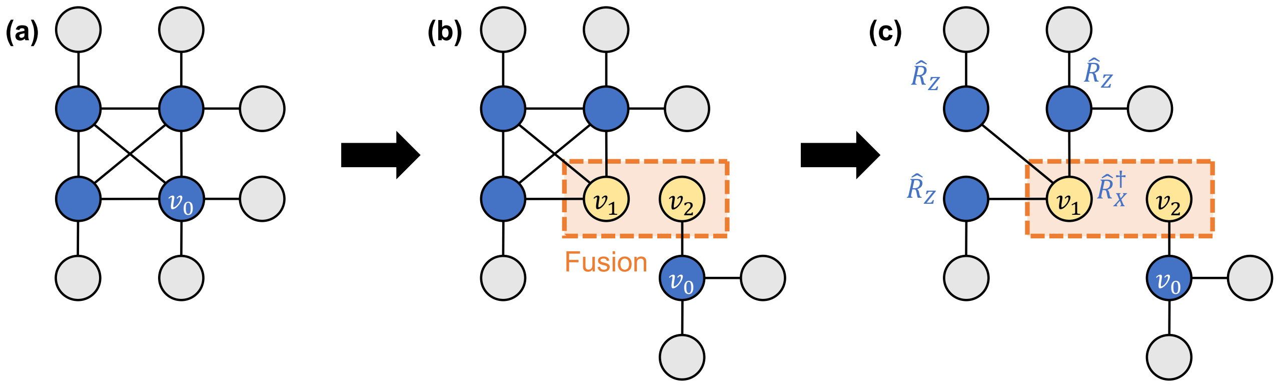

Figure 5: **Example of an unraveling process of a maximal clique.** The process is done through three steps: **(a)** selecting a random vertex v_0 in the clique, **(b)** separating v_0 from the clique by adding two vertices v_1 and v_2, and **(c)** applying a local complementation with respect to v_1. Vertices in the clique are colored in blue and the new vertices are colored in yellow. The original graph state can be constructed by performing single-qubit Clifford operations ($\hat{R}_P := \exp\left[-i(\pi/4)\hat{P}\right]$ for $\hat{P} \in \left\{\hat{X}, \hat{Z}\right\}$) and a fusion on the unraveled graph state in **(c)**.

3. If $V_{\text{no.outer}}$ is empty, do:
   (a) Select a vertex v_0 randomly from V_{cl}.
   (b) Separate v_0 and its neighbors that are not in V_{cl} from the clique and add a new vertex v_1 at the original position of v_0.
   (c) Add another new vertex v_2 and connect it with v_0.
   (d) Add (v_1, v_2) to $\mathcal{F}$.
4. Transform the graph by a local complementation with respect to v_1 and update $\hat{U}_{\text{C}} \leftarrow \hat{U}_{\text{LC}}(v_1)\hat{U}_{\text{C}}$.

In our strategy, we repeat the cycle of finding non-overlapping maximal cliques (that do not share any vertices) and unraveling them as above until no new cliques are found. Listing all maximal cliques of a graph is an important problem in the graph theory and known to take exponential time in the worst case [48]. However, there exist algorithms to list them in polynomial time per clique [49, 50], thus the problem can be efficiently solved if the graph does not contain many cliques. Our Python package *OptGraphState* uses `Graph.maximal_clique` method from Python package *python-igraph* [51], which implements a more advanced algorithm in Ref. [52].

2.1.3 Additional notes

It is a subtle problem which of bipartitely-complete graphs and cliques to unravel first. We randomly choose it since we currently have no basis for judging which one is better.

One may expect that BCSs and cliques are quite non-trivial and not very common. However, the smallest BCS and clique that are concerned for unraveling are cycles with four vertices and with three vertices, respectively. These are simple enough and appear in various graphs such as square and triangular grid graphs and RHG lattices. Moreover, large bipartitely-complete subgraphs may appear when converting a logically encoded graph state into a graph state with physical qubits. For instance, a three-qubit linear graph state with the (n, m) parity encoding contains at most (n, m) bipartitely-complete subgraphs in the physical level; see Sec. 3.3 and Ref. [35] for more details.

2.2 Construction of fusion network

After unraveling a graph G, we obtain an unraveled graph G_{unrv} along with the information $\left(\hat{U}_{\text{C}}, \mathcal{F}\right)$ that identifies the operations needed to to restore $|G\rangle$ from $|G_{\text{unrv}}\rangle$. In particular, the fusions specified by $\mathcal{F}$ are called *external fusions* to distinguish them from the fusions used to generate the unraveled graph state. We now deal with the problem of building a fusion network from result.

We first formally define fusion networks as follows:

Definition 2.3 (**Fusion network**)**.** A graph $\mathcal{N}_f = (N, L)$ is a *fusion network* of a graph state $|G\rangle$ (where vertices and edges are refereed to as *nodes* and *links*) for *root indicators* $\{r_{l,n} \in \{0,1\} \mid \forall l \in L,\ \forall n \in l\}$ if $|G\rangle$ can be generated by the process:

1. Prepare a state $\left|G_*^{(3)}\right\rangle$ for each node n. Let $q_{\text{root}}^{(n)}$ denote its root qubit and $Q_{\text{leaf}}^{(n)}$ denote the set of its leaf qubits.
2. For each link $l = \{n_1, n_2\}$, iterate the following:
   (a) Let q_1 be $q_{\text{root}}^{(n_1)}$ if $r_{l,n_1} = 1$ and an arbitrary unmeasured qubit in $Q_{\text{leaf}}^{(n_1)}$ if $r_{l,n_1} = 0$. Define q_2 analogously for n_2.
   (b) Apply appropriate single-qubit Clifford operations on q_1 and q_2, if required.
   (c) Perform a fusion on q_1 and q_2.
3. Apply appropriate single-qubit Clifford operators on the remaining qubits, if required.

We say that a link $l = \{n_1, n_2\}$ has the type of *root-to-root*, *root-to-leaf*, or *leaf-to-leaf*, when both r_{l,n_1} and r_{l,n_2} are equal to 1, only one of them is equal to 1, and both of them are equal to 0, respectively.

We now describe how to build a fusion network $\mathcal{N}_f$ and the corresponding root indicators $\{r_{l,n}\}$ from the unraveled graph G_{unrv} and the external fusions $\mathcal{F}$. The main idea is to decompose a graph state into multiple star graph state, each of which is again decomposed into multiple $\left|G_*^{(3)}\right\rangle$ states.

An m-qubit star graph state $\left|G_*^{(m)}\right\rangle$ can be constructed by conducting fusions on $m-2$ copies of $\left|G_*^{(3)}\right\rangle$, which leads to a fusion network with $m-2$ nodes connected linearly with root-to-leaf links; see Fig. 6(a) for an example when $m = 5$. Note that there is an ambiguity in positioning the root qubit (marked as "R") of $\left|G_*^{(m)}\right\rangle$ as depicted in Fig. 6(a) with (1) and (2). The node for the $\left|G_*^{(3)}\right\rangle$ state containing the root qubit of $\left|G_*^{(m)}\right\rangle$ is called the *seed node* (marked as "S") of the *node group* that consists of these $m-2$ nodes.

A general graph state $|G\rangle$ can be generated by conducting fusions on leaf qubits of multiple star graph states, where each star graph is originated from a vertex in G with degree larger than one. Consequently, its fusion network can be constructed by connecting the fusion networks of the individual star graphs (which respectively form one node group) with leaf-to-leaf links. An example is illustrated in Fig. 6(b), where root-to-leaf links and leaf-to-leaf links are represented by black single lines and orange double lines, respectively. If an external fusion exists, it also creates a link (blue dashed line) between nodes of different star graphs, as shown in Fig. 6(b). Such a link may belong to any one of the three types,

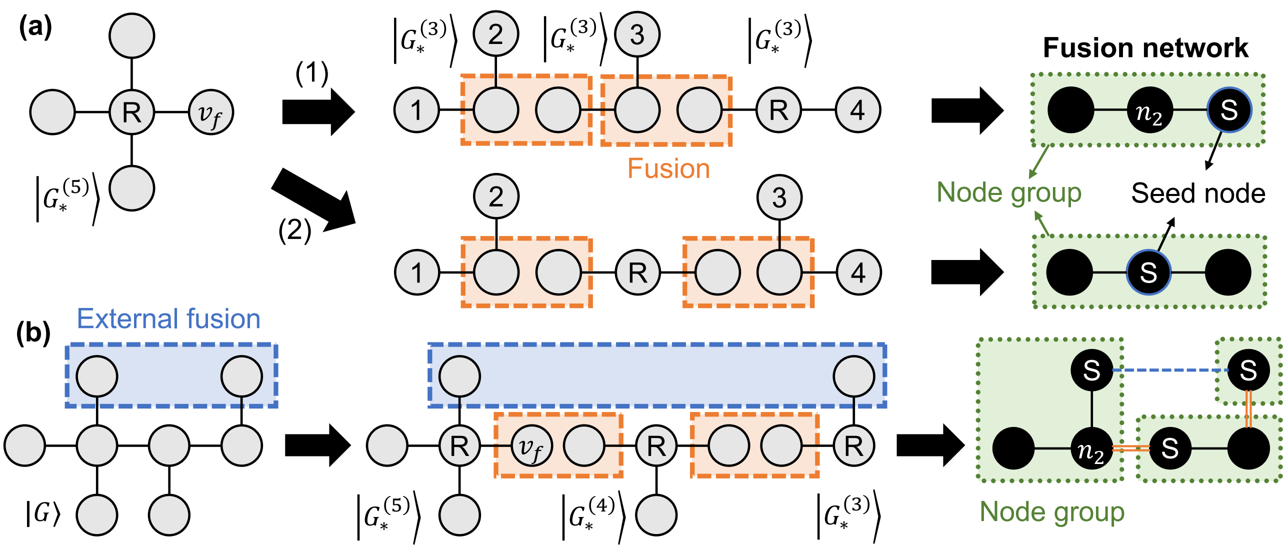

Figure 6: **Examples of the construction of fusion networks.** **(a)** A five-qubit star graph state $\left|G_*^{(5)}\right\rangle$ and **(b)** a general graph state $|G\rangle$ are considered. In **(a)**, $\left|G_*^{(5)}\right\rangle$ is decomposed into three $\left|G_*^{(3)}\right\rangle$ states, which leads to a three-node linear fusion network that forms one node group. The process varies depending on the selection of the seed node (marked as "S"), which determines the root vertex of $G_*^{(5)}$ (marked as "R"). A leaf vertex v_f of the star graph can be any of the four vertices (1–4) after the decomposition. In **(b)**, $|G\rangle$ is decomposed into multiple star graph states, where each of them is again decomposed into $\left|G_*^{(3)}\right\rangle$ states and forms one node group in the fusion network. The line styles of the links in the fusion network indicate their origins: black solid lines for fusions inside a star graph, orange double lines for fusions between star graphs, and a blue dashed line for an external fusion.

depending on the vertices involved in the external fusion. Note that external fusions always appear between different star graphs, considering the unraveling processes in Figs. 4 and 5.

It is important that the above process contains two types of ambiguity for each star graph (which are determined randomly in our strategy): which node in the node group to select as its seed node and which node to include each leaf vertex in. To illustrate the latter factor with the example of Fig. 6(a), the leaf vertex v_f in $G_*^{(5)}$ can be any of the four vertices (1–4) after the decomposition. Such ambiguity matters if $G_*^{(5)}$ appears during the decomposition of a larger graph state. In other words, if v_f participates in a fusion, the resulting fusion network may vary depending on this selection. For example, $G_*^{(5)}$ appears in the decomposition of Fig. 6(b), and in this case, vertex 3 in (1) is selected to be v_f; thus, the link for the fusion is connected to node n_2.

We lastly note that the single-qubit Clifford operators required in the process of generating a graph state can be identified from $\hat{U}_{\mathrm{C}}$, the product of single-qubit Clifford operations obtained from the unraveling process, and the fusion outcomes.

2.3 Determination of fusion order

We now have one stage left: how to determine the order of fusions. Let us regard a fusion network as a weighted graph where each node indicates a group of entangled qubits and each link represents a fusion between these groups that needs to be done. The weight of each node $w(n)$, which is initialized to 1, is defined as the resource overhead of the process of generating the corresponding entangled states. Namely, $w(n)$ is the average number of required $\left|G_*^{(3)}\right\rangle$ states to generate the state.

Upon the above setting, the action of a fusion can be treated as the *contraction* of a link l, which means to merge two endpoints (n_1, n_2) of the link into a new node n_l and reconnect the links connected to the original nodes with n_l. The weight of n_l is updated as

$$w(n_l) := \begin{cases} [w(n_1) + w(n_2)]/p_{\text{succ}} & \text{if } n_1 \neq n_2, \\ w(n_1)/p_{\text{succ}} & \text{if } n_1 = n_2, \end{cases} \tag{5}$$

where p_{succ} is the fusion success probability. Hence, if the order of the fusions is given, the resource overhead Q of the entire process can be obtained as the summation of the weights of the last remaining nodes after contracting all the edges in the order. Note that an intermediate fusion network during this process may have loops (links connecting a node to itself) or multi-links (links incident to the same two vertices).

For a fusion network $\mathcal{N}_\text{f} = (N, L)$, the number of possible fusion orders is $|L|!$; thus it is extremely inefficient to randomly sample fusion orders and find the best one unless there are very few links. Instead of it, our strategy is based on the following two intuitions:

1. It is preferred to contract links with small weights first, where the weight of a link l is defined as $w(n_l)$ in Eq. (5). It is because, defining $f(x, y) := (x + y)/p_{\text{succ}}$ for two numbers x and y,
$$f(f(w_1, w_2), w_3) < f(w_1, f(w_2, w_3))$$
when $w_1 < w_2 < w_3$.

2. Links that do not share endpoints can be contracted simultaneously and it is preferred to contract links as parallelly as possible. For example, let us consider a four-node linear fusion network where the node set is $\{n_1, n_2, n_3, n_4\}$ and n_i and n_{i+1} are connected with a link $l_{i,i+1}$ for each $i \in \{1, 2, 3\}$. Provided that $p_{\text{succ}} = 1/2$, we obtain $Q = 16$ if $l_{1,2}$ and $l_{3,4}$ are first contracted in parallel, but obtain $Q = 22$ if $l_{2,3}$ is contracted first.

Based on these intuitions, we introduce the *min-weight-maximum-matching-first* method to determine the fusion order. For each round of the link contraction process, we first identify the set of links with the smallest weight and get the subgraph $\mathcal{N}_{\text{min.wgt}}$ of the intermediate fusion network induced by these links. We then find a maximum matching of $\mathcal{N}_{\text{min.wgt}}$, which is the largest set of links that do not share any nodes, and contract these links in parallel. By repeating this procedure until no links remain, we can determine the fusion order and calculate the resource overhead Q. We illustrate an example in Fig. 7. To compute a maximum matching, our software uses `max_weight_matching` function from Python package *NetworkX* [53], which

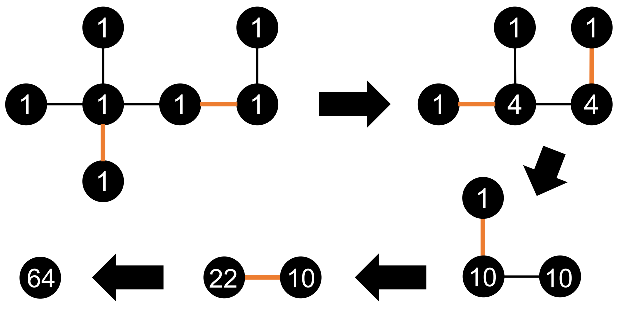

Figure 7: **Example of the determination of the fusion order with the min-weight-maximum-matching-first method.** We assume $p_{\text{succ}} = 1/2$. Each step is an intermediate fusion network after contracting links (orange bold lines) in the previous step. The numbers inside the nodes indicate their weights. The obtained resource overhead is $Q = 64$, which is the weight of the last remaining node.

is based on the algorithm in Ref. [54] and takes time of $\mathcal{O}\left(|\text{number of nodes}|^3\right)$.

2.3.1 Note on the average number of fusions

One may want to use the average number of required fusion attempts to quantify resource overheads instead of the average number of required $\left|G_*^{(3)}\right\rangle$ states. In such a case, Eq. (5) should be modified to

$$w(n_l) := \begin{cases} [w(n_1) + w(n_2) + 1]/p_{\text{succ}} & \text{if } n_1 \neq n_2, \\ [w(n_1) + 1]/p_{\text{succ}} & \text{if } n_1 = n_2, \end{cases}$$

and the weights of nodes should be initialized to 0. All the other parts of the strategy remain the same. *OptGraphState* provides an option to use this alternative resource measure instead of Q.

2.4 Iteration

Since the method introduced above has randomness in several stages, it may produce a different outcome each time it is attempted. We thus iterate the method a sufficient number of times and choose the best one.

Our software uses an *adaptive* method to determine the iteration number: Denoting the process to iterate the strategy m times as $R(m)$, we first perform $R(m_{\text{init}})$ for a given integer $m_{\text{init}} \geq 1$ and obtain $Q_{\text{opt}}^{(1)}$, which is the minimal value of Q obtained from the samples. We then perform $R(2m_{\text{init}})$ and obtain $Q_{\text{opt}}^{(2)}$. If $Q_{\text{opt}}^{(1)} \leq Q_{\text{opt}}^{(2)}$, we stop the iteration and return $Q^{(1)}_{\text{opt}}$. If otherwise, we perform $R(4m_{\text{init}})$, obtain $Q^{(3)}_{\text{opt}}$, and stop the iteration if $Q^{(2)}_{\text{opt}} \leq Q^{(3)}_{\text{opt}}$, which returns $Q^{(2)}_{\text{opt}}$. If $Q^{(2)}_{\text{opt}} > Q^{(3)}_{\text{opt}}$, we perform $R(8m_{\text{init}})$, obtain $Q^{(4)}_{\text{opt}}$, and so on. Here, we refer to this method as the *adaptive iteration* with the given value of m_{init}.

We emphasize that our strategy does not guarantee that the obtained generation method is strictly optimal. As elaborated above, our approach involves simulating sufficiently many random samples and selecting the best performing one.

3 Applications of the strategy

In this section, we present the numerical results obtained by applying our strategy to various graphs. We first analyze the distribution of resource overheads for random graphs, showing its tendency with respect to the numbers of vertices and edges. We then provide numerical evidence indicating that each step of our strategy can significantly contribute to lowering the resource overhead. Additionally, we show the calculated resource overheads of various well-known graphs described in Sec. 1.1. We lastly investigate the probability of successfully generating a graph state given a restricted number of available basic resource states and present the results for several graph states.

Throughout the section, $|V|$ and $|E|$ for a given graph indicate the numbers of vertices and edges, respectively, and $|E|_{\max}$ is defined as the maximal possible number of edges for a given value of $|V|$ (under the assumption that there are no loops and multi-edges): $|E|_{\max} = |V|(|V|-1)/2$.

3.1 Analysis of random graphs

To sample random graphs, we use the Erdős–Rényi model [55], where all graphs that contain given fixed values of $|V|$ and $|E|$ have an equal probability. Figure 8 visualizes the distributions of the obtained resource overheads optimized by our strategy for various values of $|V|$ and $|E|$ when $p_{\text{succ}} = 0.5$ or 0.75. Here, we sample 100 random graphs for each combination $(p_{\text{succ}}, |V|, |E|)$ and use the adaptive iteration method of $m_{\text{init}} = 600$. Several observations from the results are as follows:

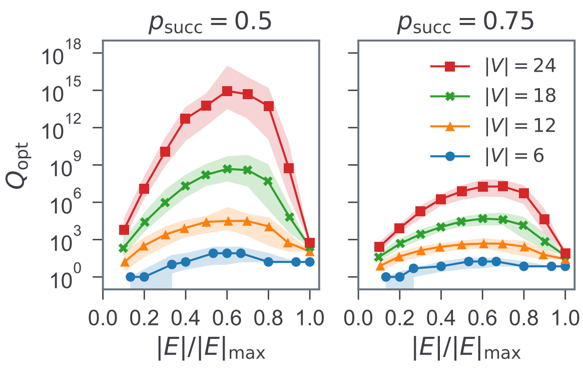

Figure 8: **Distribution of the optimized resource overhead Q_{opt} for random graphs.** Random graphs are sampled with fixed numbers of vertices ($|V|$) and edges ($|E|$) by the Erdős–Rényi model [55]. Two different fusion success rates are considered: $p_{\text{succ}} \in \{0.5, 0.75\}$. $|E|_{\max} = |V|(|V|-1)/2$ is the maximal possible number of edges for the given vertex number. For each combination of $(p_{\text{succ}}, |V|, |E|)$, we sample 100 random graphs and obtain the distribution of Q_{opt} through the adaptive iteration method of $m_{\text{init}} = 600$. The median of the distribution is indicated as a dot and its total range is shown as a shaded region.

- Q_{opt} increases exponentially (or super-exponentially) as $|V|$ grows when $|E|/|E|_{\max}$ is fixed.
- For a fixed value of $|V|$, Q_{opt} is maximal when $|E| \approx 0.6|E|_{\max}$. Q_{opt} is inversely correlated with $|E|$ for large values of $|E|$ since bipartitely-complete subgraphs and cliques are more likely to appear for when $|E|$ is large.
- The fusion scheme with $p_{\text{succ}} = 0.75$ may greatly reduce the order of Q_{opt}, compared to the one with $p_{\text{succ}} = 0.5$, especially when $|V|$ is large. Note that, to achieve $p_{\text{succ}} = 0.75$ with linear optics, we require an ancillary two-photon Bell state [23] or four ancillary unentangled photons [24] per fusion and photon-number resolving detectors that can discriminate at least four photons. On the other hand, the scheme with $p_{\text{succ}} = 0.5$ requires only on-off detectors and no ancillary photons.

3.2 Performance analysis

We now show that our strategy is indeed effective by comparing it with two "deficient" strategies in which a certain stage is missing from the original

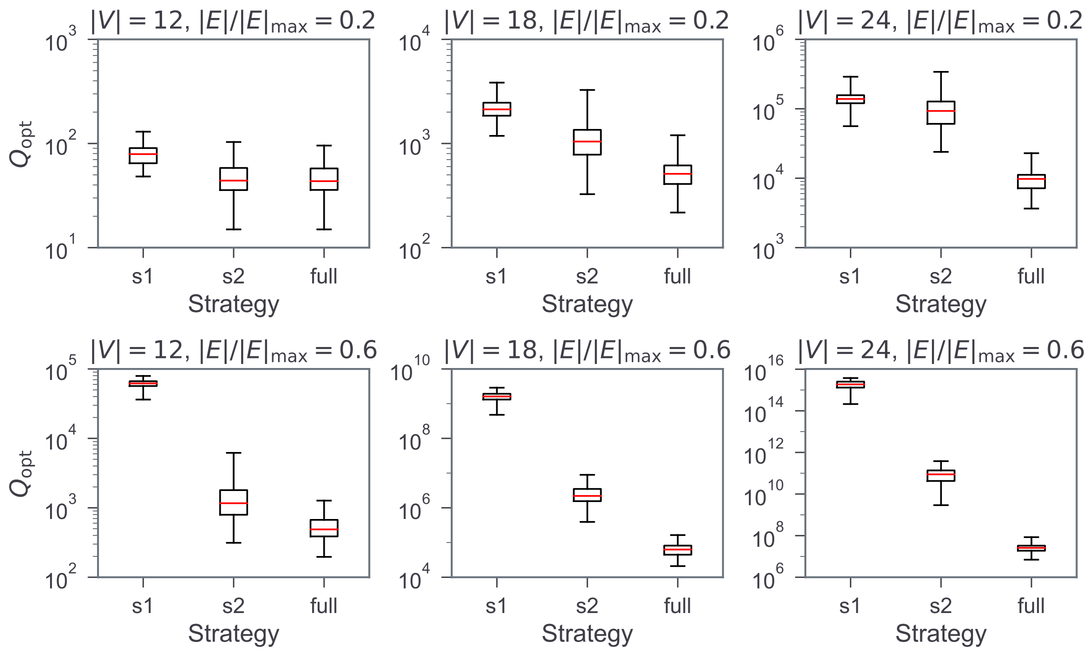

Figure 9: **Comparison of the distributions of the optimized resource overhead Q_{opt} for different optimization strategies.** Three strategies are considered: the strategy without unraveling (s1), the strategy with random selection of the fusion order (s2), and our full strategy. The subplots correspond to different values of $|V| \in \{12, 18, 24\}$ and $|E|/|E|_{\max} \in \{0.2, 0.6\}$. For each setting, 100 graphs are sampled by the Erdős–Rényi model and 1200 iterations are done for each graph. (The adaptive iteration method is not used for fair comparisons.) The distribution of Q_{opt} is visualized as a box plot, where the red line indicates the median, the box extends from the first quartile (Q1) to the third quartile (Q3), and the whisker covers the entire range of the values.

"full" strategy. In detail, we consider the following two alternative strategies:

(s1) The strategy without the unraveling process, where the original graph is directly used for generating a fusion network. The other steps are the same as the full strategy.

(s2) The strategy where the fusion order is randomly selected without using the min-weight-maximum-matching-first method. The other steps are the same as the full strategy.

In Fig. 9, the distributions of Q_{opt} optimized by these three strategies for random graphs are presented as box plots. Each box extends from the first quartile (Q1) to the third quartile (Q3) and the corresponding whisker covers the entire range of the values. It clearly shows that the full strategy is significantly more powerful than the deficient ones, especially when there exist many vertices and edges. In other words, each step in the full strategy contributes to reducing the resource overhead.

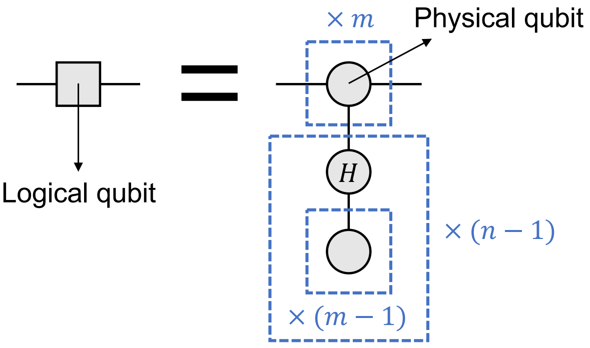

Figure 10: **Rule for converting an (n, m) parity-encoded graph state into a physical-level graph state.** A dashed box with a text "$\times N$" (for an integer N) indicates a bundle of recurrent subgraphs. Namely, the subgraph inside the box is repeated N times and, for each edge crossing the border of the box, an edge of the same format exists for every repeated subgraph. See Ref. [35] for more details.

3.3 Applications to well-known graphs

We here investigate the resource overheads of the graph states in Sec. 1.1, which are utilized in various quantum tasks such as MBQC, FBQC, quantum repeaters, and quantum error correction. Be-

Table 1: **Results of resource overhead analyses for various well-known graphs.** See Fig. 2 for the visualization of these graphs. $|V|$ and $|E|$ indicate the numbers of vertices and edges. $|E|_{\max} = |V|(|V|-1)/2$ is the maximal possible number of edges. The optimized resource overheads Q_{opt} and the corresponding average numbers of fusion attempts are calculated for two fusion success rates: $p_{\text{succ}} \in \{0.5, 0.75\}$. The adaptive iteration method of $m_{\text{init}} = 1200$ is used for the calculation.

Graph	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert E\rvert/\lvert E\rvert_{\max}$	$p_{\text{succ}} = 0.5$		$p_{\text{succ}} = 0.75$	
				Q_{opt}	#Fusions	Q_{opt}	#Fusions
6-vertex star ($G_*^{(6)}$)	6	5	0.33	1.6×10^1	1.0×10^1	7.1×10^0	4.9×10^0
12-vertex star ($G_*^{(12)}$)	12	11	0.17	1.1×10^2	7.4×10^1	2.7×10^1	2.1×10^1
18-vertex star ($G_*^{(18)}$)	18	17	0.11	2.6×10^2	1.7×10^2	5.1×10^1	4.0×10^1
24-vertex star ($G_*^{(24)}$)	24	23	0.083	5.4×10^2	3.6×10^2	8.2×10^1	6.5×10^1
$(3,3)$-lattice	9	12	0.33	5.4×10^2	3.7×10^2	5.5×10^1	4.6×10^1
$(4,4)$-lattice	16	24	0.20	7.7×10^3	5.2×10^3	2.4×10^2	2.0×10^2
$(5,5)$-lattice	25	40	0.13	1.0×10^5	6.7×10^4	9.9×10^2	8.2×10^2
$(6,6)$-lattice	36	60	0.095	7.9×10^5	5.3×10^5	2.8×10^3	2.3×10^3
$(1,1,1)$-RHG lattice	18	24	0.16	1.9×10^4	1.3×10^4	3.9×10^2	3.4×10^2
$(2,2,2)$-RHG lattice	90	144	0.036	2.8×10^{13}	1.8×10^{13}	8.0×10^6	6.5×10^6
$(2,2)$-tree	7	6	0.29	2.8×10^1	1.8×10^1	1.0×10^1	7.3×10^0
$(2,2,2)$-tree	15	14	0.13	2.1×10^2	1.4×10^2	4.0×10^1	3.1×10^1
$(2,2,2,2)$-tree	31	30	0.065	1.6×10^3	1.1×10^3	1.4×10^2	1.1×10^2
$(3,3,3)$-tree	40	39	0.050	1.7×10^3	1.2×10^3	1.8×10^2	1.5×10^2
$(4,4,4)$-tree	85	84	0.024	1.2×10^4	7.8×10^3	6.1×10^2	4.9×10^2
$(8,2,2)$-tree	57	56	0.035	1.6×10^4	1.0×10^4	4.7×10^2	3.8×10^2
Repeater graph with $m=3$	12	21	0.32	1.2×10^2	8.2×10^1	2.8×10^1	2.1×10^1
Repeater graph with $m=4$	16	36	0.30	2.1×10^2	1.4×10^2	4.3×10^1	3.3×10^1
Repeater graph with $m=6$	24	78	0.28	5.4×10^2	3.6×10^2	8.2×10^1	6.5×10^1
$(2,2)$ parity-encoded 3-star	12	17	0.26	1.2×10^2	8.2×10^1	2.8×10^1	2.1×10^1
$(3,3)$ parity-encoded 3-star	27	48	0.14	8.8×10^2	5.9×10^2	1.1×10^2	8.4×10^1
$(4,4)$ parity-encoded 3-star	48	95	0.084	2.4×10^3	1.6×10^3	2.3×10^2	1.9×10^2
$(5,5)$ parity-encoded 3-star	75	158	0.057	1.0×10^4	6.8×10^3	5.3×10^2	4.3×10^2
$(2,2)$ parity-encoded 6-cycle	24	42	0.15	1.3×10^3	8.5×10^2	1.2×10^2	9.9×10^1
$(3,3)$ parity-encoded 6-cycle	54	114	0.080	6.7×10^3	4.4×10^3	3.9×10^2	3.1×10^2
$(4,4)$ parity-encoded 6-cycle	96	222	0.049	2.1×10^4	1.4×10^4	8.8×10^2	7.1×10^2

sides them, we also consider parity-encoded graph states, which are used for basic resource states of parity-encoding-based topological quantum computing (PTQC) protocol in Ref. [35] and FBQC in Ref. [6]. The (n,m) *parity code* (or *generalized Shor code*) [56] encodes a single logical qubit with the basis of

$$\left\{\left(|0\rangle^{\otimes m} + |1\rangle^{\otimes m}\right)^{\otimes n} \pm \left(|0\rangle^{\otimes m} - |1\rangle^{\otimes m}\right)^{\otimes n}\right\},$$

where $\{|0\rangle, |1\rangle\}$ is the physical-level basis. An (n,m) *parity-encoded graph state* indicates a graph state in which the qubits on the vertices are encoded with the (n,m) parity code. Such an encoded graph state can be rewritten as a graph state of physical-level qubits according to the rule in Fig. 10 [35]. We cover two types of logical-level graphs in the calculation: the 3-vertex star graph (for PTQC [35]) and 6-vertex cycle graph (for FBQC [6]).

In Table 1, we list the results of the resource analyses for these graph states, together with the basic information of the graphs. Additionally, in Appendix A, we present several explicit examples of the application of our strategy with visualization. We note that the extremely high resource overheads of RHG lattices do not necessarily render them impractical. From the perspective of fault-tolerant quantum computing [4], certain levels of absent vertices or edges resulting from fusion failures can be endured [57, 58]. Hence, many previous schemes [21, 39, 35, 6] take an approach of initially generating "unit" resource states successfully, followed by merging them while allowing fusion failures. In this context, our strategy can be utilized to evaluate the resource costs of these unit states. For example, they can be tree graphs [21], parity-encoded 3-star graphs [39, 35], or parity-encoded 6-cycle graphs [6], all of which are presented in Table 1.

3.4 Success probability of graph state generation

We emphasize that the resource overhead Q we have used so far is defined by the *expected value* of the resource count required to generate the desired graph state $|G\rangle$. Namely, if we define a discrete random variable C by this resource count, then

$$Q = \mathbb{E}[C] = \sum_{c=1}^{\infty} c \Pr(C = c).$$

One may want to know more information on the probability mass function (PMF) $\Pr(C = c)$, not just its expectation value. Moreover, the corresponding cumulative mass function (CMF) indicates the probability $P_{\mathrm{succ}}(c)$ of the successful generation of $|G\rangle$ when c resource states are provided, namely,

$$P_{\mathrm{succ}}(c) = \sum_{c'=1}^{c} \Pr\left(C = c'\right), \tag{6}$$

which may be a more practical indicator than Q to assess the performance of a scheme.

We first investigate the case where two graph states with resource counts C_1 and C_2 are merged by a single fusion of success rate p_{s} to form a graph state with resource count C_3. For each $i \in \{1, 2, 3\}$, let us define $q_i(\cdot)$ by the probability distribution function (PDF) corresponding to the PMF $\Pr\left(C_i = c\right)$; namely,

$$q_i(x) := \sum_{c=1}^{\infty} \Pr\left(C_i = c\right)\delta(x - c),$$

where δ is the Kronecker delta function. Then q_1, q_2, and q_3 are related as

$$\begin{aligned} q_3 &= p_{\mathrm{s}}(q_1 * q_2) + p_{\mathrm{s}}(1-p_{\mathrm{s}})(q_1 * q_1 * q_2 * q_2) \\ &+ p_{\mathrm{s}}(1-p_{\mathrm{s}})^2(q_1 * q_1 * q_1 * q_2 * q_2 * q_2) \\ &+ p_{\mathrm{s}}(1-p_{\mathrm{s}})^3(q_1 * q_1 * q_1 * q_1 * q_2 * q_2 * q_2 * q_2) \\ &+ \cdots, \end{aligned} \tag{7}$$

where the "$*$" symbol indicates convolution defined as

$$(f * g)(x) := \int_{-\infty}^{\infty} f(t)g(x-t)\mathrm{d}t.$$

Here, each term including $p_{\mathrm{s}}(1-p_{\mathrm{s}})^l$ for $l \geq 0$ corresponds to the case where the fusion fails l times and succeeds on the next attempt. The convolution theorem [59] states that the Fourier transformation (FT) defined as

$$\mathcal{F}[f]\,(k) = \tilde{f}(k) := \int_{-\infty}^{\infty} f(x)e^{ikx}\mathrm{d}x$$

converts convolution into multiplication of functions as

$$\mathcal{F}[f * g]\,(k) = \tilde{f}(k)\tilde{g}(k).$$

Thus, applying the FT to the both sides of Eq. (7), we obtain

$$\begin{aligned} \tilde{q}_3(k) &= \sum_{l=0}^{\infty} p_{\mathrm{s}}(1-p_{\mathrm{s}})^l[\tilde{q}_1(k)\tilde{q}_2(k)]^{l+1} \\ &= \frac{p_{\mathrm{s}}\tilde{q}_1(k)\tilde{q}_2(k)}{1-(1-p_{\mathrm{s}})\tilde{q}_1(k)\tilde{q}_2(k)}, \end{aligned}$$

which gives

$$\tilde{q}_3(k)^{-1} = \frac{1}{p_{\mathrm{s}}}\tilde{q}_1(k)^{-1}\tilde{q}_2(k)^{-1} - \frac{1-p_{\mathrm{s}}}{p_{\mathrm{s}}}. \tag{8}$$

Therefore, considering that the PDF of the resource count for the basic resource state $\left|G_*^{(3)}\right\rangle$ is $\delta(x-1) =: q_{\mathrm{base}}(x)$, the Fourier-transformed PDF (FTPDF) for every graph state can be written as the inverse of a polynomial of

$$\tilde{q}_{\mathrm{base}}^{-1}(k) = e^{-ik} =: z^{-1}.$$

Namely, for every FTPDF $\tilde{q}(k)$, there exists a series of real numbers $\{a_l\}_{l=0}^{L}$ that satisfies

$$\tilde{q}(k) = \frac{1}{\sum_{l=0}^{L} a_l z^{-l}} = \frac{1}{\sum_{l=0}^{L} a_l e^{-ikl}}. \tag{9}$$

To compute the FTPDF for a desired graph state, we just need to assign the PDF of z to every node of the fusion network and sequentially apply the rule of Eq. (8) for every link contraction.

After obtaining the final FTPDF $\tilde{q}(k)$ in the form of Eq. (9), we need to recover the PMF $\Pr(C = c)$ from it. By applying the Taylor expansion at $z = 0$ on

$$\tilde{q}(k) = \frac{z^L}{\sum_{l=0}^{L} a_l z^{L-l}} = \frac{z^L}{\sum_{l=0}^{L} a_{L-l} z^l},$$

we get the power series

$$\tilde{q}(k) = z^L \sum_{j=0}^{\infty} b_j z^j,$$

where $\{b_j\}$ is recurrently defined as

$$b_0 = a_L^{-1},$$
$$b_j = - \sum_{j'=\max(0,j-L)}^{j-1} \frac{a_{L-j+j'}}{a_L} b_{j'}.$$

Since $z^j = e^{ikj}$ is the FT of $\delta(x-j)$, we conclude that the PMF is

$$\Pr(C=c) = \begin{cases} 0 & \text{if } c < L, \\ b_{c-L} & \text{otherwise.} \end{cases}$$

Additionally, we obtain the CMF $P_{\text{succ}}(c)$ in Eq. (6) as

$$P_{\text{succ}}(c) = \begin{cases} 0 & \text{if } c < L, \\ d_{c-L} & \text{otherwise,} \end{cases}$$

where

$$d_0 = a_L^{-1},$$
$$d_j = \sum_{j'=\max(0,j-L-1)}^{j-1} \frac{a_{L-j+j'+1} - a_{L-j+j'}}{a_L} d_{j'},$$

if we define $a_{-1} := 0$.

The above calculations are implemented in our software *OptGraphState*. In Fig. 11, we display the computed success probabilities P_{succ} for generating various graph states, plotted against the resource count c. Lines labeled (a)–(d) correspond to the repeater graph with $m = 6$, $(4,4)$ parity-encoded 3-star graph, $(4,4)$-lattice graph, and $(4,4)$ parity-encoded 6-cycle graph, respectively. We note that the resource overhead Q, which is shown as dashed lines, matches the resource count when the associated success probability is roughly 60%. For achieving a greater success probability, such as 90%, one would need approximately twice as many resource states as Q.

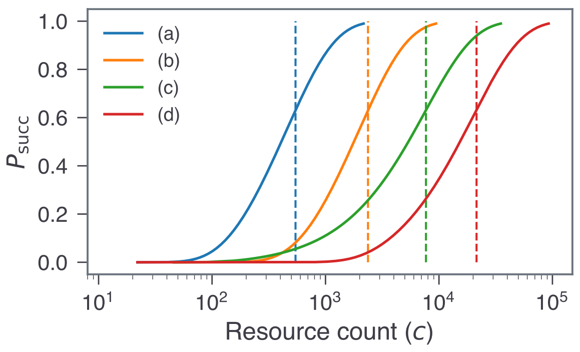

Figure 11: **Success probabilities of graph state generation as functions of the number of provided basic resource states.** The solid lines respectively indicate the success probabilities $P_{\text{succ}}(c)$ for the graph states of **(a)** the repeater graph with $m = 6$, **(b)** $(4,4)$ parity-encoded 3-star, **(c)** $(4,4)$-lattice, and **(d)** $(4,4)$ parity-encoded 6-cycle. Each dashed line represents the corresponding resource overhead Q, which is the expected value of the resource count when P_{succ} is the cumulative mass function.

4 Remarks

Graph states are versatile resource states for various tasks on quantum computation and communication, such as MBQC [2, 4], FBQC [6], quantum error correction [7, 8], and quantum repeaters [11]. However, in optical systems, the non-deterministic nature of entangling operations hinders the generation of large-scale graph states; thus, the generation process should be carefully designed.

In this work, we introduced a graph-theoretical strategy to construct a resource-efficient method for generating an arbitrary graph state with the type-II fusion operation. Here, the resource overhead is quantified by the average number of required basic resource states (three-qubit star graph states) to generate the graph state without failed fusions. As outlined in Sec. 2, the strategy is composed of multiple trials to find the optimal one, where each round contains three stages: unraveling the graph, constructing a fusion network, and determining the fusion order. In Sec. 3, we applied the strategy to various graph states and verified numerically that each step of the strategy is indeed necessary to achieve high resource efficiency. Moreover, we described a recursive technique to determine the success probability of generating a graph state as a function of the resource cost and tested it on several representative graph states.

We anticipate that our strategy and software will aid researchers in designing experimentally feasible approaches utilizing photonic graph states and in evaluating the practicality of their proposed schemes. For example, the basic resource states of MBQC and FBQC can be logically-encoded star or cycle graph states [35, 6]. Employing larger or more complex codes may improve the fault-tolerance of these schemes; however, generating such resource states could become a bottleneck in their implementation. Our strategy can contribute to evaluating such a trade-off relation and identifying the most practical sweet spot.

We lastly note several interesting unsolved problems related to our work:

1. **Generalization of unraveling.** For a given graph state $|G\rangle$, how can we identify another graph state $|G'\rangle$ such that $|G\rangle$ can be generated from $|G'\rangle$ using a combination of fusions, single-qubit Clifford (or general) operations, single-qubit measurements, and classical communications, resulting in a reduction of the overall resource overhead? This problem bears similarities to the equivalence problem of graph states [38, 60, 61], but fusions are included as allowable operations and resource overheads for fusion-based generation are considered.

2. **Lower bound of resource overhead.** Is it possible to find a (sufficiently tight) lower bound of the resource overhead Q? If such a lower bound can be computed, it would enable us to assess whether the resource overhead optimized by our strategy is indeed close to the real optimal value.

3. **Behavior of Q_{opt} against $|E|/|E|_{\max}$.** In Fig. 8, Q_{opt} exhibits an intriguing behavior, where it is maximized around $|E|/|E|_{\max} = 0.6$ regardless of $|V|$. Can it be explained analytically? Is Q_{opt} related to a specific property of the graph or graph state, such as the multipartite entanglement of the graph state [62]?

4. **Usage of larger basic resource states.** Using a graph state larger than the three-qubit star graph state $\left|G_*^{(3)}\right\rangle$ as the basic resource state can be beneficial. While preparing larger basic resource state might be challenging, it can reduce the resource overhead. We anticipate that the reduction would be approximately proportional to the original overhead of the new basic resource state. For instance, employing $\left|G_*^{(4)}\right\rangle$ could lead to a fourfold reduction in overhead. If $\left|G_*^{(n)}\right\rangle$ is used as the basic resource state, only minor adjustments are needed in our strategy: Within each node group of a fusion network shown in Fig. 6, contract adjacent nodes beforehand so that each node represents a star graph state with at most n qubits. However, if the basic resource state is not a star graph state, it might necessitate a complete overhaul of the algorithm, which will be worth investigating.

5. **Tolerance for fusion failures.** The aim of our strategy is to generate a graph state without any failed fusions. However, in practical scenarios, we may consider allowing some fusion failures at the cost of some missing vertices and edges from the lattice, which can be tolerable depending on the characteristics and usage of the generated state. For example, in parity-encoded graph states, which are graph states of logically encoded qubits, such defects may lead to correctable errors. If we allow a degree of failed fusions, how many vertices or edges would be missing from the resulting lattice? In such cases, how can we determine an efficient generation scheme and calculate its resource overhead? Can we relate the fault-tolerance of a graph state and the resource overhead to generate it?

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grants funded by the Korean government (Grant Nos. NRF-2020R1A2C1008609, NRF-2023R1A2C1006115, and NRF-2022M3K4A1097117) via the Institute of Applied Physics at Seoul National University, by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (IITP-2021-0-01059 and IITP-2023-2020-0-01606).

References

[1] M. Hein, W. Dür, J. Eisert, R. Raussendorf, M. Van den Nest, and H.-J. Briegel. "Entanglement in graph states and its appli-

cations". In Quantum Computers, Algorithms and Chaos. Pages 115–218. IOS Press (2006).

[2] Robert Raussendorf and Hans J. Briegel. "A one-way quantum computer". Phys. Rev. Lett. **86**, 5188–5191 (2001).

[3] Robert Raussendorf, Daniel E. Browne, and Hans J. Briegel. "Measurement-based quantum computation on cluster states". Phys. Rev. A **68**, 022312 (2003).

[4] R. Raussendorf, J. Harrington, and K. Goyal. "A fault-tolerant one-way quantum computer". Ann. Phys. **321**, 2242–2270 (2006).

[5] R. Raussendorf, J. Harrington, and K. Goyal. "Topological fault-tolerance in cluster state quantum computation". New J. Phys. **9**, 199 (2007).

[6] Sara Bartolucci, Patrick Birchall, Hector Bombin, Hugo Cable, Chris Dawson, Mercedes Gimeno-Segovia, Eric Johnston, Konrad Kieling, Naomi Nickerson, Mihir Pant, et al. "Fusion-based quantum computation". Nat. Commun. **14**, 912 (2023).

[7] D. Schlingemann and R. F. Werner. "Quantum error-correcting codes associated with graphs". Phys. Rev. A **65**, 012308 (2001).

[8] A. Pirker, J. Wallnöfer, H. J. Briegel, and W. Dür. "Construction of optimal resources for concatenated quantum protocols". Phys. Rev. A **95**, 062332 (2017).

[9] Damian Markham and Barry C. Sanders. "Graph states for quantum secret sharing". Phys. Rev. A **78**, 042309 (2008).

[10] B. A. Bell, Damian Markham, D. A. Herrera-Martí, Anne Marin, W. J. Wadsworth, J. G. Rarity, and M. S. Tame. "Experimental demonstration of graph-state quantum secret sharing". Nat. Commun. **5**, 5480 (2014).

[11] M. Zwerger, W. Dür, and H. J. Briegel. "Measurement-based quantum repeaters". Phys. Rev. A **85**, 062326 (2012).

[12] M. Zwerger, H. J. Briegel, and W. Dür. "Universal and optimal error thresholds for measurement-based entanglement purification". Phys. Rev. Lett. **110**, 260503 (2013).

[13] Koji Azuma, Kiyoshi Tamaki, and Hoi-Kwong Lo. "All-photonic quantum repeaters". Nat. Commun. **6**, 6787 (2015).

[14] J. Wallnöfer, M. Zwerger, C. Muschik, N. Sangouard, and W. Dür. "Two-dimensional quantum repeaters". Phys. Rev. A **94**, 052307 (2016).

[15] Nathan Shettell and Damian Markham. "Graph states as a resource for quantum metrology". Phys. Rev. Lett. **124**, 110502 (2020).

[16] Michael A. Nielsen. "Optical quantum computation using cluster states". Phys. Rev. Lett. **93**, 040503 (2004).

[17] Daniel E. Browne and Terry Rudolph. "Resource-efficient linear optical quantum computation". Phys. Rev. Lett. **95**, 010501 (2005).

[18] Jeremy C. Adcock, Sam Morley-Short, Joshua W. Silverstone, and Mark G. Thompson. "Hard limits on the postselectability of optical graph states". Quantum Sci. Technol. **4**, 015010 (2018).

[19] Holger F. Hofmann and Shigeki Takeuchi. "Quantum phase gate for photonic qubits using only beam splitters and postselection". Phys. Rev. A **66**, 024308 (2002).

[20] T. C. Ralph, N. K. Langford, T. B. Bell, and A. G. White. "Linear optical controlled-NOT gate in the coincidence basis". Phys. Rev. A **65**, 062324 (2002).

[21] Ying Li, Peter C. Humphreys, Gabriel J. Mendoza, and Simon C. Benjamin. "Resource costs for fault-tolerant linear optical quantum computing". Phys. Rev. X **5**, 041007 (2015).

[22] Samuel L. Braunstein and A. Mann. "Measurement of the Bell operator and quantum teleportation". Phys. Rev. A **51**, R1727–R1730 (1995).

[23] W. P. Grice. "Arbitrarily complete Bell-state measurement using only linear optical elements". Phys. Rev. A **84**, 042331 (2011).

[24] Fabian Ewert and Peter van Loock. "3/4-efficient Bell measurement with passive linear optics and unentangled ancillae". Phys. Rev. Lett. **113**, 140403 (2014).

[25] Seung-Woo Lee, Kimin Park, Timothy C. Ralph, and Hyunseok Jeong. "Nearly deterministic Bell measurement with multiphoton entanglement for efficient quantum-information processing". Phys. Rev. A **92**, 052324 (2015).

[26] Seung-Woo Lee, Timothy C. Ralph, and Hyunseok Jeong. "Fundamental building

block for all-optical scalable quantum networks". Phys. Rev. A **100**, 052303 (2019).

[27] Keisuke Fujii and Yuuki Tokunaga. "Fault-tolerant topological one-way quantum computation with probabilistic two-qubit gates". Phys. Rev. Lett. **105**, 250503 (2010).

[28] Ying Li, Sean D. Barrett, Thomas M. Stace, and Simon C. Benjamin. "Fault tolerant quantum computation with nondeterministic gates". Phys. Rev. Lett. **105**, 250502 (2010).

[29] H. Jeong, M. S. Kim, and Jinhyoung Lee. "Quantum-information processing for a coherent superposition state via a mixedentangled coherent channel". Phys. Rev. A **64**, 052308 (2001).

[30] H. Jeong and M. S. Kim. "Efficient quantum computation using coherent states". Phys. Rev. A **65**, 042305 (2002).

[31] Srikrishna Omkar, Yong Siah Teo, and Hyunseok Jeong. "Resource-efficient topological fault-tolerant quantum computation with hybrid entanglement of light". Phys. Rev. Lett. **125**, 060501 (2020).

[32] Srikrishna Omkar, Y. S. Teo, Seung-Woo Lee, and Hyunseok Jeong. "Highly photon-loss-tolerant quantum computing using hybrid qubits". Phys. Rev. A **103**, 032602 (2021).

[33] Shuntaro Takeda, Takahiro Mizuta, Maria Fuwa, Peter Van Loock, and Akira Furusawa. "Deterministic quantum teleportation of photonic quantum bits by a hybrid technique". Nature **500**, 315–318 (2013).

[34] Hussain A. Zaidi and Peter van Loock. "Beating the one-half limit of ancilla-free linear optics Bell measurements". Phys. Rev. Lett. **110**, 260501 (2013).

[35] Seok-Hyung Lee, Srikrishna Omkar, Yong Siah Teo, and Hyunseok Jeong. "Parity-encoding-based quantum computing with bayesian error tracking". npj Quantum Inf. **9**, 39 (2023).

[36] Gerald Gilbert, Michael Hamrick, and Yaakov S. Weinstein. "Efficient construction of photonic quantum-computational clusters". Phys. Rev. A **73**, 064303 (2006).

[37] Konrad Kieling, David Gross, and Jens Eisert. "Minimal resources for linear optical one-way computing". J. Opt. Soc. Am. B **24**, 184–188 (2007).

[38] 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).

[39] Srikrishna Omkar, Seok-Hyung Lee, Yong Siah Teo, Seung-Woo Lee, and Hyunseok Jeong. "All-photonic architecture for scalable quantum computing with greenberger-horne-zeilinger states". PRX Quantum **3**, 030309 (2022).

[40] Michael Varnava, Daniel E. Browne, and Terry Rudolph. "Loss tolerance in one-way quantum computation via counterfactual error correction". Phys. Rev. Lett. **97**, 120501 (2006).

[41] N. Lütkenhaus, J. Calsamiglia, and K.-A. Suominen. "Bell measurements for teleportation". Phys. Rev. A **59**, 3295–3300 (1999).

[42] 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).

[43] C. Schön, E. Solano, F. Verstraete, J. I. Cirac, and M. M. Wolf. "Sequential generation of entangled multiqubit states". Phys. Rev. Lett. **95**, 110503 (2005).

[44] Netanel H. Lindner and Terry Rudolph. "Proposal for pulsed on-demand sources of photonic cluster state strings". Phys. Rev. Lett. **103**, 113602 (2009).

[45] 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).

[46] Shuntaro Takeda, Kan Takase, and Akira Furusawa. "On-demand photonic entanglement synthesizer". Science Advances **5**, eaaw4530 (2019).

[47] 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).

[48] John W. Moon and Leo Moser. "On cliques in graphs". Isr. J. Math. **3**, 23–28 (1965).

[49] Eugene L. Lawler, Jan Karel Lenstra, and A. H. G. Rinnooy Kan. "Generating all maximal independent sets: NP-hardness and

polynomial-time algorithms". SIAM J. Comput. **9**, 558–565 (1980).

[50] Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. "A new algorithm for generating all the maximal independent sets". SIAM J. Comput. **6**, 505–517 (1977).

[51] Gabor Csardi and Tamas Nepusz. "The igraph software package for complex network research". InterJournal **Complex Systems**, 1695 (2006). url: https://igraph.org.

[52] David Eppstein, Maarten Löffler, and Darren Strash. "Listing all maximal cliques in sparse graphs in near-optimal time". In International Symposium on Algorithms and Computation. Pages 403–414. Springer (2010).

[53] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. "Exploring network structure, dynamics, and function using NetworkX". In Gäel Varoquaux, Travis Vaught, and Jarrod Millman, editors, Proceedings of the 7th Python in Science Conference (SciPy2008). Pages 11–15. Pasadena, CA USA (2008). url: https://www.osti.gov/biblio/960616.

[54] Zvi Galil. "Efficient algorithms for finding maximum matching in graphs". ACM Comput. Surv. **18**, 23–38 (1986).

[55] Paul Erdős and Alfréd Rényi. "On random graphs I". Publicationes mathematicae **6**, 290–297 (1959).

[56] T. C. Ralph, A. J. F. Hayes, and Alexei Gilchrist. "Loss-tolerant optical qubits". Phys. Rev. Lett. **95**, 100501 (2005).

[57] Sean D. Barrett and Thomas M. Stace. "Fault tolerant quantum computation with very high threshold for loss errors". Phys. Rev. Lett. **105**, 200502 (2010).

[58] James M. Auger, Hussain Anwar, Mercedes Gimeno-Segovia, Thomas M. Stace, and Dan E. Browne. "Fault-tolerant quantum computation with nondeterministic entangling gates". Phys. Rev. A **97**, 030301 (2018).

[59] G. B. Arfken, H. J. Weber, and F. E. Harris. "Mathematical methods for physicists: A comprehensive guide". Elsevier Science. (2011). url: https://books.google.co.kr/books?id=JOpHkJF-qcwC.

[60] 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).

[61] Axel Dahlberg and Stephanie Wehner. "Transforming graph states using single-qubit operations". Philos. T. Roy. Soc. A **376**, 20170325 (2018).

[62] M. Hein, J. Eisert, and H. J. Briegel. "Multiparty entanglement in graph states". Phys. Rev. A **69**, 062311 (2004).

A Examples of the application of the strategy

In this Appendix, we present examples of applying our strategy to several graphs, which is obtained by using our Python package *OptGraphState*. Figures 12, 13, and 14 show the $(4,4)$*-lattice graph*, *repeater graph with* $m = 4$, and $(2,2)$ *parity-encoded 6-cycle graph* and their unraveled graphs and fusion networks that give the resource overheads in Table 1 when $p_{\mathrm{succ}} = 0.5$, which are plotted by using the *python-igraph* library. The description of the various elements of these figures is as follows.

Original graph:

- A number inside each vertex is its unique name.
- Orange vertices indicate qubits with non-trivial Clifford gates.

Unraveled graph:

- A number inside each vertex is its unique name. If a vertex is originated from a vertex in the original graph, they have the same name.
- Black solid lines are edges of the unraveled graph and red dashed lines indicate external fusions.
- Orange vertices indicate qubits with non-trivial Clifford gates.

Fusion network:

- A number inside each node is its unique name. Each seed node has the same name as the vertex in the unraveled graph that the node is originated from; namely, the qubit at the vertex is the root qubit of the $\left|G_*^{(3)}\right\rangle$ state of the seed node. Non-seed nodes have names like 'i-j', where i is the name of the seed node in the same node group and j is an index starting from 1.
- Black solid lines are leaf-to-leaf links, red dashed lines are root-to-root links, and blue arrows are root-to-leaf links. Each arrow for a root-to-leaf link points from the node that contains the leaf qubit involved in the fusion to the other node.
- A number placed on each link indicates the order of the fusion. The fusions should be performed in the order in which these numbers increase and those with the same number can be done simultaneously.
- If the text placed on a link ends with the letter 'C' (such as '1C'), it means that the corresponding fusion is accompanied by non-trivial Clifford gates applied to one or both of the qubits before the fusion is performed.

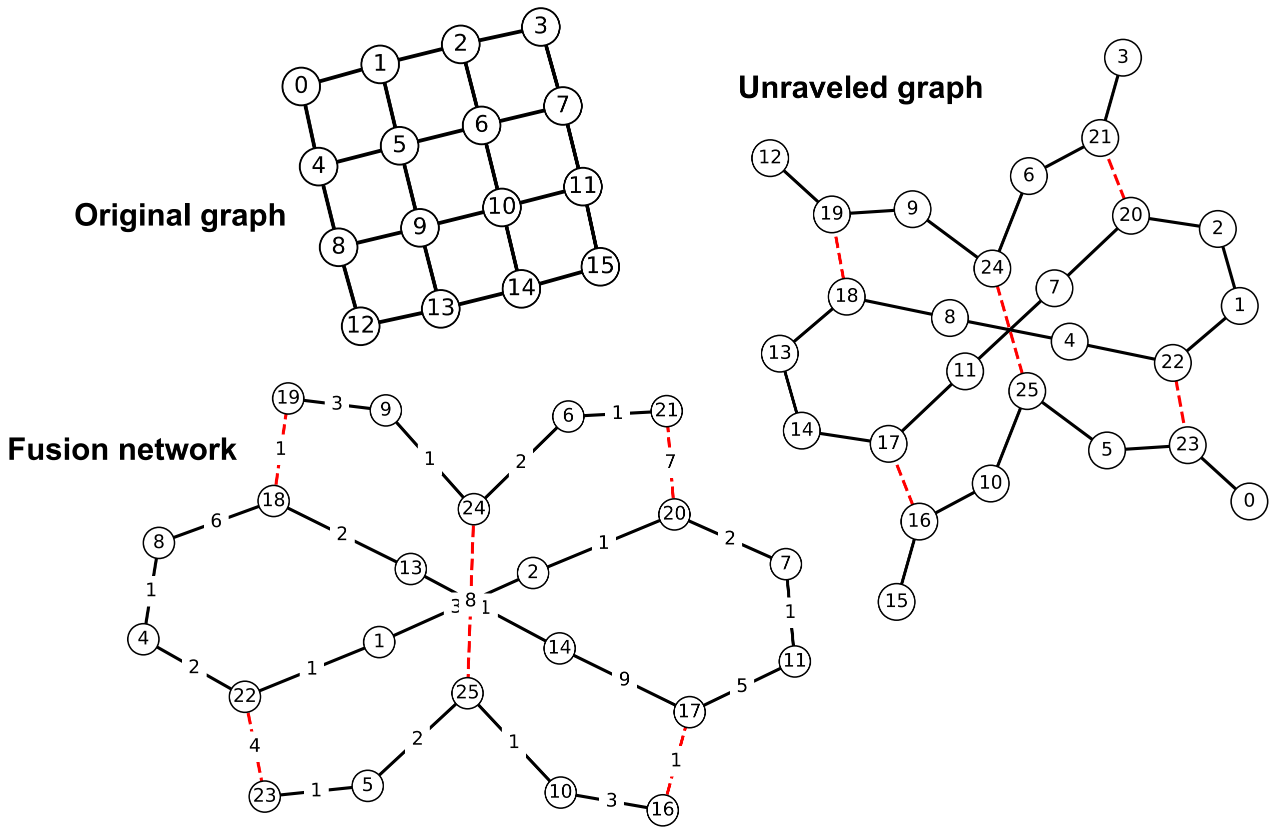

Figure 12: $(4, 4)$**-lattice graph and its unraveled graph and fusion network obtained by our strategy.** The unraveled graph and fusion network give the resource overhead of 7680 when $p_{\text{succ}} = 0.5$. See Appendix A for their interpretation.

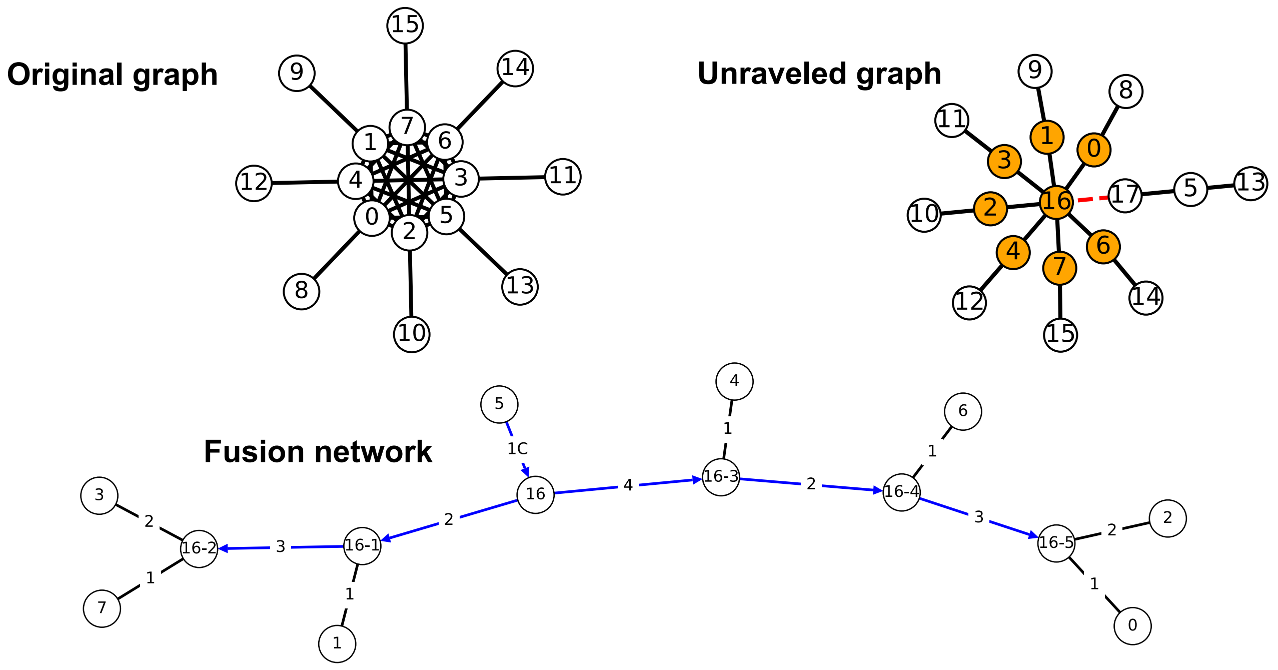

Figure 13: **Repeater graph with** $m = 4$ **and its unraveled graph and fusion network obtained by our strategy.** The unraveled graph and fusion network give the resource overhead of 208 when $p_{\text{succ}} = 0.5$. See Appendix A for their interpretation. In the unraveled graph, the qubits at the orange vertices are subjected to non-trivial Clifford gates: $R_X(\pi/2) := \exp\left[i(\pi/4)\hat{X}\right]$ to qubit 16 and $R_Z(\pi/2) := \exp\left[i(\pi/4)\hat{Z}\right]$ to the others. In the fusion network, the fusion of the link connecting the nodes 5 and 16 is accompanied by $R_X(\pi/2)$ applied to the qubit in node 16 before the fusion is performed.

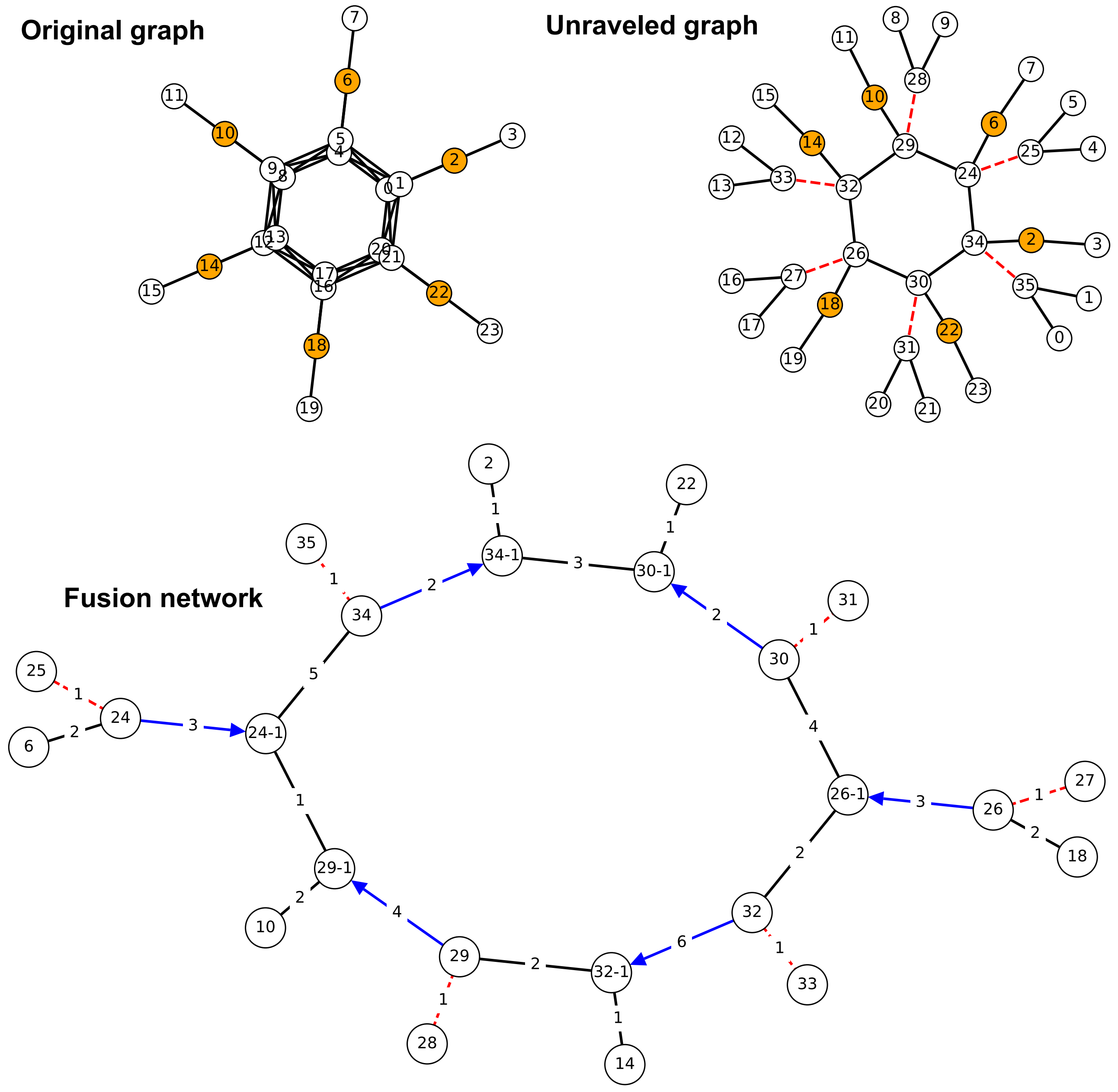

Figure 14: $(2, 2)$ **parity-encoded 6-cycle graph and its unraveled graph and fusion network obtained by our strategy.** The unraveled graph and fusion network give the resource overhead of 1280 when $p_{\text{succ}} = 0.5$. See Appendix A for their interpretation. In the original and unraveled graphs, the qubits at the orange vertices are subjected to the Hadamard gate.