Efficient percolation simulations for lossy photonic fusion networks

Matthias C. Löbl [1,*] Stefano Paesani,[1,2] and Anders S. Sørensen [1]

[1]*Center for Hybrid Quantum Networks (Hy-Q), The Niels Bohr Institute, University of Copenhagen, Blegdamsvej 17,*
DK-2100 Copenhagen Ø, Denmark

[2]*NNF Quantum Computing Programme, The Niels Bohr Institute, University of Copenhagen, Blegdamsvej 17,*
DK-2100 Copenhagen Ø, Denmark.

The study of percolation phenomena has various applications ranging from social networks or materials science to quantum information. The most common percolation models are bond or site percolation for which the Newman-Ziff algorithm enables an efficient simulation. Here, we consider several nonstandard percolation models that appear in the context of measurement-based photonic quantum computing with so-called graph states and fusion networks. The associated percolation thresholds determine the tolerance to photon loss in such systems and we develop modifications of the Newman-Ziff algorithm to efficiently perform the corresponding percolation simulations. We demonstrate our algorithms by using them to characterize exemplary fusion networks and graph states.

I. INTRODUCTION

In the field of quantum information, graph states are quantum states that can be represented by a graph: vertices are qubits, the fundamental storage unit of quantum information, and graph edges represent quantum mechanical entanglement between the qubits [1]. Such graph states are the key ingredient for measurement-based quantum computing [2,3]. A suitable large graph state (a so-called cluster state) provides a universal resource for this quantum computing paradigm [2,3], allowing quantum computation to be performed by doing measurements on the qubits. This approach is particularly suitable for photonic implementations of quantum information processing, where qubits are encoded in the optical degrees of freedom of single photons and processed through linear optical interferometers [4–7]. It is, however, challenging to realize a sufficiently large graph state of photons because, first, it is nontrivial to create edges/entanglement between photons in the graph state, and, second, photons can be lost with a non-negligible probability p_{loss}. Both of these limitations lead to imperfect graphs with some level of randomness, but tools from graph theory such as path-finding methods [8,9] and percolation models [5,10,11] can be employed to convert the imperfect graph state into a useful resource for quantum computing.

To create graph edges, probabilistic entangling operations [4,12] are typically considered for photonic qubits as deterministic photon-photon gates are difficult to realize [13]. These can be implemented by joint two-qubit measurements called *fusions* or *Bell state measurements*, which consume the two measured photons (graph nodes). The desired measurement outcomes lead to successful fusions which, loosely speaking, make a connection between two graphs. Other measurement results herald fusion failure which does not generate the connection. Using fusions, many small graph states (named *resource states*) can be connected to create larger entangled states [5,11,14]. If a large entangled graph state is generated by such a process, it can be converted into a resource for measurement-based quantum computing through a process called lattice renormalization [9]. Whether applying many fusion operations leads to such a large graph state depends on the probability of fusion success, p_s, and the connectivity of all fusions. In the absence of loss, the required value for p_s can correspond to a bond percolation threshold λ_{bond} of the *fusion network* determined by the geometric arrangement of the fusions [5,10,11]. If $p_s < \lambda_{bond}$, the resulting graph state consists of small pieces that are not useful for quantum computing. For $p_s > \lambda_{bond}$, a large connected graph state is created that percolates the fusion network, spanning from one of its ends to the other. In the latter case, the size of the state scales linearly with the system size, providing the desired resource for measurement-based quantum computing.

The second issue, photon loss, is problematic for practical realizations since system efficiencies $\eta = 1 - p_{loss}$ are so far significantly below unity. This problem is due to absorption or leakage inside the employed photonic circuits and especially due to the employed photon sources [15–18]. A loss leads to a mixed quantum state [19] and a pure state can only be retained by removing the neighborhood of the lost qubit (the qubits that are entangled with the lost qubit) from the graph by measurements [5,14] [see Fig. 1(a)].

In this paper, we consider probabilistic fusion networks in the presence of loss. In these fusion networks, small resource states are fused to make a large entangled state. We show that the required values for the efficiency η correspond to
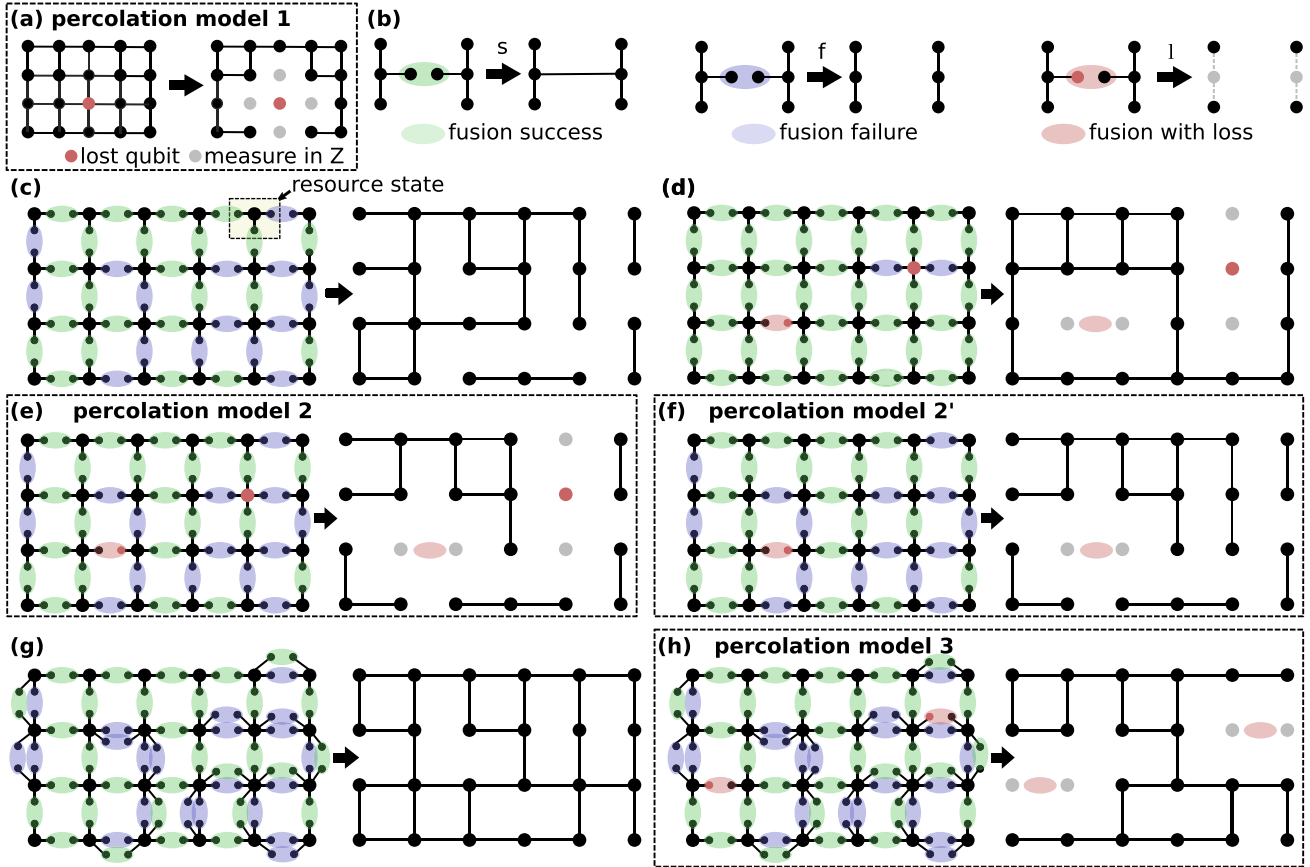

FIG. 1. The different percolation models. (a) Percolation model 1: loss on an existing graph state. Photon loss is compensated by measuring the neighborhood of a lost qubit in the Z basis, causing the deletion of its entire neighborhood. (b) Fusion of photons from two different graph states. The fusion can be realized by sending the two fusion photons (encircled by ellipses) to a suitable Bell state measurement setup [4,5,14]. The two photons undergoing the Bell state measurement are removed from the graph, yet the measurement may lead to new connections being added to the graph depending on the measurement outcome. If the fusion succeeds (s), the neighbor of one fusion photon gets connected to the neighbor of the other one. If the fusion fails (f), there is no connection. If one or more fusion photons are lost, the neighborhoods of both qubits involved in the fusion need to be measured in the Z basis, removing these nodes from the graph (gray). (c) Graph state generation from small resource states. Many star-shaped resource states are arranged on a square lattice and fused (for simplicity illustrated without photon loss). After the fusions, a new graph state is generated. To be of use for measurement-based quantum computing, the fusion success probability p_s must be above the bond-percolation threshold of the fusion lattice. (d) Loss of a fusion photon/leaf qubit and loss of a central qubit. Loss of a fusion photon results in the deletion of the central qubit of two star-shaped resource states (left). If a central qubit of a star-shaped resource state is lost, all central qubits that are connected to it by a successful fusion need to be deleted (right). For this illustration, we have assumed that most fusions succeed. (e) Percolation model 2: photon losses and fusion failure happen at the same time. In this percolation model, there are losses of leaf and central qubits, as illustrated. (f) Percolation model 2′: only leaf qubits can be lost. (g) Example of a repeat-until-success scheme where failed fusions are repeated up to $n_{max} = 2$ fusions in total, by generating two new fusion photons. (h) Percolation model 3: Repeat-until-success scheme with photon loss.

percolation thresholds λ_η of various new percolation models. To generate a graph state that is useful for measurement-based quantum computing, the efficiency must be above λ_η, which thus determines the loss tolerance of the fusion network. The percolation model corresponding to the process of photon loss differs from standard percolation models (such as bond or site percolation [20], or a combination of both [21]) as photon loss is more destructive than a missing site in a site-percolation model. Therefore, new algorithms are required to efficiently simulate the corresponding percolation models. We extend the algorithm developed by Newman and Ziff [22,23] for bond and site percolation such that it can be used to efficiently perform these new percolation simulations.

We consider several percolation models: first, a given graph state suffers photon loss (percolation model 1). Second, many *star*-shaped resource states are arranged on a periodic fusion network. Fusions are applied between *fusion photons* on the *leaf* nodes of neighboring resource states to connect them to a larger graph. We consider both purely photonic resource states (percolation model 2) as well as resource states where the central qubit is the spin of a quantum emitter, e.g., atom [24] or an equivalent solid-state system [25–30] (percolation model 2′). Finally, we consider repeat-until-success fusion networks where quantum emitters can generate new fusion photons conditioned on the failure of previous fusions (percolation model 3).

We have implemented all algorithms in C and the code is publicly available as an open-source repository [31]. We recently used this source code to simulate various lattices of several dimensions and found that their loss tolerances can strongly differ [14]. Notably, photonic approaches are not restricted to a particular spatial arrangement of the qubits. This makes it possible to create lattices with various connection patterns and this can be used to increase the robustness to photon loss.

II. GRAPH STATES

Within quantum information science, graph states are represented by a graph $G = (V, E)$ corresponding to a quantum state where the vertices of the graph, V, are the qubits and the edges, E, are entangling operations between the qubits. The graph state can be defined as

$$|G\rangle = \prod_{(i,j)\in E} C_{ij} |+\rangle^{\otimes V}, \qquad (1)$$

where $C_{ij} = (|00\rangle\langle00| + |01\rangle\langle01| + |10\rangle\langle10| - |11\rangle\langle11|)_{ij}$ is a controlled-Z gate (controlled phase gate) between qubits i and j, $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ is an equal superposition of the two basis states $|0\rangle$ and $|1\rangle$, and $|+\rangle^{\otimes V}$ represents the product state where all qubits V are in this state [1]. The order in which the controlled-Z gates C_{ij} are applied is irrelevant as all gates C_{ij} commute. Therefore, a controlled-Z gate can be represented by an undirected graph edge $e_{ij} = (i, j)$, and the graph state can be represented by the set of edges. Since the controlled-Z operation is an entangling operation when applied to two qubits in the $|+\rangle$ state, the edges represent the entanglement structure of the state. Graph states are part of both the larger classes of stabilizer states [32] and the class of hypergraph states [33]. For more detailed information on graph states, we refer the reader to Ref. [1].

Single-qubit measurements can be used to manipulate graph states [1] and use them for universal quantum computing [2]. For this manuscript, it is only necessary to consider the effect of Z-basis measurements that project the measured qubit on the computational basis $\{|0\rangle, |1\rangle\}$ (the Z basis). To understand the effect on the graph state, we pick a qubit $A \in V$ and write Eq. (1) as

$$\prod_{k\in N(A)} C_{Ak} |+\rangle_A |\theta\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A |\theta\rangle + |1\rangle_A Z_{N(A)} |\theta\rangle), \qquad (2)$$

where $|\theta\rangle = \prod_{(i,j)\in E, i\neq A\neq j} C_{ij} |+\rangle^{\otimes V\setminus A}$ is the graph state corresponding to the induced subgraph where node A has been removed from the graph G. $N(A) = \{k \in V \mid \exists (A, k) \in E\}$ represents all qubits in the neighborhood of A, with $Z_{N(A)}$ representing a Pauli-Z gate ($Z = |0\rangle\langle0| - |1\rangle\langle1|$) on all these qubits. Assume one measures qubit A in the Z basis. Projecting on the first Z eigenstate, $|0\rangle_A$, yields the state $|\theta\rangle$ and thus corresponds to removing qubit A and its edges from the graph. Projecting on $|1\rangle_A$ leads to the state $Z_{N(A)} |\theta\rangle$ corresponding to qubit A being removed and a Pauli-Z gate applied to all its neighbors from before the measurement [1]. Since such single-qubit gates do not change the entanglement properties, the resulting state has the same computational power as one without the gates applied and can typically be dealt with

by reinterpreting the outcomes of later measurements. For this article, the outcomes of the Z-basis measurements are thus inconsequential and will be ignored, although one will have to keep track of the outcome in an actual experimental implementation.

We consider photonic graph states where one of the main imperfections is the loss of qubits. Loss of qubit A can be interpreted as measuring the qubit without knowing the outcome of the measurement, i.e., without knowing whether or not the gates $Z_{N(A)}$ have been applied to the remaining state $|\theta\rangle$. The resulting quantum state is thus a mixed state. To retain a pure quantum state, the qubits in $N(A)$ need to be removed from the graph by measuring them in the Z basis [5] [see Fig. 1(a)].[1]

III. DEFINITION OF THE PERCOLATION MODELS

In this section, we will define the percolation models, for which we will develop efficient algorithms.

A. Percolation model 1: Loss on graph states

In the first model, we only consider the effect of loss on an already existing graph state (no fusions), where each qubit is represented by two states of a single photon, e.g., two different polarization states. Assume a given graph state $|G\rangle$ represented by a graph $G = (V, E)$. We only consider graphs that can be embedded as some periodic lattice, but this is, in principle, not necessary. Assume that $|G\rangle$ suffers photon loss with a rate of $p_{\text{loss}} = 1 - \eta$. As described before, the neighborhood of every lost photon needs to be measured in the Z basis, resulting in the neighbors being removed from the graph state. This corresponds to a modified site-percolation model where a missing site leads to a deletion of its entire neighborhood [see Fig. 1(a)]. The corresponding percolation threshold λ_η specifies the fraction of photons (graph nodes from V) that at least need to be present (not lost) such that in the limit of infinite lattice size, there is a connected graph state left (some induced subgraph of G) which has a size that scales linearly with the size of G. Such a large graph state is required for measurement-based quantum computing and the loss tolerance of the graph state $|G\rangle$ is therefore characterized by λ_η, where $\eta = 1 - p_{\text{loss}}$ has to be above this number. Such a percolation model gives a guideline for the loss tolerance of a given graph state, yet there are weaknesses regarding its practical applicability: first, photon loss is typically unknown (unheralded) before doing a destructive measurement [5,8], yet the required lattice renormalization (see Sec. III D) relies on knowing the losses before measuring the qubits in a certain basis. Without quantum nondemolition detection of whether

---

[1]This removes the same number $|N(A)| + 1$ of qubits and so-called stabilizer generators [32] from the state. Instead, one could extract a maximum number of stabilizer generators without support on the lost qubit by Gaussian elimination and continue with the resulting mixed state. This would typically erase fewer stabilizer generators. If the considered lattice has no fault-tolerant properties, it is, however, unclear how to subsequently use the generated mixed state and we will not pursue this possibility any further.

or not a qubit is present [34], it is therefore unclear to which degree the lossy graph state is eventually useful for quantum computing. Second, it is unclear how the large initial photonic graph state can be physically created.

B. Percolation models 2, 2′: Loss on fusion network

To alleviate the shortcomings of the model considered in the previous section, we now switch to a model that includes the graph state generation. Fusion networks represent an approach to generate large graph states from smaller resource states that can more easily be generated experimentally.[2] We consider star-shaped resource states and we assume that fusions [4] of photons at the leaf nodes of the star-shaped resource states, the *fusion photons*, are used to establish connections between the *central* qubits of different star-shaped resource states.[3] Such a fusion network is illustrated in Fig. 1(c). As all the measurements associated with the fusions commute and the fusions performed are fixed (no adaptiveness), all fusions can be performed at once (ballistically [5]). We only consider fusion networks where the resource states and the fusions can be arranged on a graph G with a periodic embedding, the *fusion lattice*. A node of G represents a central qubit of a resource state and an edge e_{ij} represents a fusion of two leaf nodes connected to the central qubits i and j. The exemplary fusion network in Fig. 1(c) corresponds to a fusion lattice that is a square lattice, but other arbitrary lattice geometries are possible.

Fusions can be implemented using linear optics elements and photon detectors, with the measured photon pattern on the detectors heralding the successful implementation of the fusion [4,46]. Fusions are probabilistic, succeeding with a probability of p_s. We assume so-called rotated type-II fusion that consumes two fusion photons and connects the two central nodes of two star-shaped resource states upon success [5,14,46]. The fusion fails with probability $1 - p_s$, in which case the resource states are left mutually unconnected [see Fig. 1(b)].[4] In the absence of photon loss, a large graph state is therefore generated once p_s exceeds the bond-percolation threshold of the fusion lattice [see Fig. 1(c)] [5,10,11]. In the percolation models that we will consider in this section, p_s can, in principle, be an arbitrary probability, yet only certain values correspond to physical Bell measurement setups and we assume $p_s = 0.5$ corresponding to the most simple setup.[5]

Further, we assume that every photon, independent of whether or not it is used in a fusion, has a uniform probability

$p_{\text{loss}} = 1 - \eta$ to be lost.[6] When and where the photons are lost is generally not important as a loss will only be registered by the absence of a detection event in a measurement such as a fusion. Thus, the effect on the graph state only depends on the loss probability, regardless of where the loss is applied. When one or more photons are lost in a fusion operation, a pure state is retained by measuring the neighborhood of both fusion photons [see Fig. 1(b)] [5,14]. The left part of Fig. 1(d) illustrates the effect of a lost fusion photon on a fusion lattice in the boundary case $p_s = 1$. Losses in fusions are heralded [5,46], meaning that one knows about all lost fusion photons from the measured detection pattern, i.e., one knows that a photon is lost when it was not detected in the fusion process. A fusion with two photons attached to two central qubits i and j has three outcomes [see Fig. 1(b)]: the fusion succeeds, creating a connection between i and j (probability $p_s\eta^2$); the fusion fails to create a connection [probability $(1 - p_s)\eta^2$]; or a photon loss is detected (probability $1 - \eta^2$), in which case i, j need to be removed by Z-basis measurements (see Sec. II).

Loss of a central qubit i has a different effect, which is illustrated in the right part of Fig. 1(d): if the fusion between one of the leaf nodes of qubit i and one of the leaf nodes of qubit j succeeds, qubit j gets connected to qubit i and it thus needs to be measured in the Z basis if qubit i is lost. In the case of fusion failure, j is not connected to the lost qubit i and nothing needs to be done. It is assumed in this illustration that none of the fusion photons used for connecting qubits i, j are lost since, in this case, both i, j would need to be measured in the Z basis anyway.

The leaf qubits are measured in the fusion process such that losses are heralded. The central qubits, on the other hand, constitute the final graph state which needs to be measured in a specific basis to perform quantum computation [2]. Since the locations of the losses are typically only known when a measurement fails to detect a photon, it is unclear whether the generated graph states can be used for quantum computation in this setting [8,14]. In this case, the efficiency η being above the corresponding percolation threshold λ_η thus constitutes a necessary yet not sufficient condition for generating a graph state that is a useful resource for quantum computing. Nevertheless, we consider the possibility of all-photonic resource states as such approaches have been considered also in previous work [5,11]. We refer to the described percolation model with all-photonic resource states as percolation model 2, for which an example is given in Fig. 1(e).

A promising alternative is using resource states where the central qubit is the spin of a quantum emitter [14] and we shall refer to the corresponding model as percolation model 2′, which is illustrated in Fig. 1(f). In this case, the central qubits are not subject to loss, making the generated graph state suitable for measurement-based quantum computing provided that it is sufficiently large. This is the case once the efficiency η is above the corresponding percolation threshold λ_η of a sufficiently large fusion network. For percolation model 2′,

---

[2]Proposals for generating small resource graph states can be found in Refs. [35–37] and recent experimental realizations in Refs. [24,38–43].

[3]The considered star-shaped resource states are locally equivalent [44,45] to Greenberger-Horne-Zeilinger (GHZ) states, with $|\text{GHZ}\rangle = \frac{1}{\sqrt{2}}(|0\rangle^{\otimes n} + |1\rangle^{\otimes n})$.

[4]A more detailed explanation of the required rotated type-II fusion can be found in Fig. 4 of Ref. [5] or Fig. A5(d) of Ref. [14].

[5]The fusion success probability can be improved (boosted) above $p_s = 0.5$ by using additional photonic resources [47–49] yet $p_s = 0.5$ is easiest to implement in practice.

---

[6]A nonuniform distribution of loss probabilities may originate from different efficiencies of photon sources [16–18] or different paths that the photons take, but an investigation of this is beyond the scope of this work.

$\eta > \lambda_\eta$ thus is a necessary and sufficient condition for creating a graph state that is useful for quantum computing.

C. Percolation model 3: Adaptive fusion network

The success probability $p_s = 0.5$ of the fusions can be a limitation for the previously discussed scheme. For an implementation based on photon emitters, there is, however, a strategy to remedy this issue using adaptive fusions. We consider a scheme where the central nodes are spins in quantum emitters and fusions can be repeated on demand. To establish a connection between two emitters, each of them creates a photon entangled with it, e.g., by using a suitable optical pulse [35,37]. When the fusion succeeds, we proceed with no further actions. When it fails, it can be repeated until the fusion either succeeds, a maximum number n_{max} of fusion attempts is reached, or photon loss is detected. In Fig. 1(g), the case of $n_{max} = 2$ is illustrated in the absence of photon loss. When there are no losses, repeating failed fusions effectively leads to a boosting of the fusion success probability above $p_s = 0.5$. For the case of n_{max} in Fig. 1(g), for instance, a link between two central nodes would be generated with probability $p_s + (1 - p_s)p_s = 0.75$, which is well above the bond-percolation threshold of the fusion lattice (square lattice with $\lambda_{bond} = 0.5$ [50]). We note that there are also all-optical schemes to boost the fusion success probability [47,48,51], but the proposed repeat-until-success method is less sensitive to photon loss as additional fusion photons are only generated on demand.

As before, the percolation model becomes more involved when photon loss is present, in which case a sequence of fusion attempts has three possible outcomes: success, failure, or photon loss. In the success case, no photon loss occurs and the final fusion succeeds and thus creates a connection between two central nodes. Fusion failure (no connection) happens when all n_{max} fusion attempts fail with no photon loss. When a photon loss is detected in any of the fusion attempts, the two quantum emitters are measured in the Z basis, removing them from the graph. The combination of these possibilities of fusion outcomes represents percolation model 3, as illustrated in Fig. 1(h).

The described adaptive approach is related to well-known schemes for creating spin-spin entanglement via Bell measurements and corresponding repeat-until-success schemes [52–54]. An advantage of the considered scheme is that fusions do not influence other fusions (they are locally adaptive), which enables doing all fusions in parallel. That reduces the overhead compared to adaptive divide-and-conquer approaches [52,54,55]. Furthermore, our scheme can be applied to emitters with only a single spin per node (in contrast to, e.g., Ref. [56]).

D. Lattice renormalization

With the models described above, a large connected cluster state that scales linearly with the size of the fusion network or the initial graph state is created if the photon efficiency is above the percolation threshold λ_η. The resulting graph state does not represent a fault-tolerant lattice, which eventually is required for quantum computing [57]. The obtained cluster

state can, however, be renormalized into such a lattice by suitable single-qubit measurements [8,9]. A complete investigation of this process is beyond the scope of this article. We therefore restrict our analysis to noting that once the efficiency is above the threshold, scalable quantum computation is, in principle, possible, although different geometries may be more advantageous than others. Note, further, that it is required to know all imperfections (losses and fusion failures) before applying the measurement pattern performing the lattice renormalization [1,8]. This is fulfilled for percolation model 3 and percolation model 2′ with a central qubit that is a spin (since fusion failure and loss of fusion photons is heralded). In these cases, an efficiency $\eta = 1 - p_{loss}$ above the percolation threshold λ_η is thus a sufficient condition for generating the desired lattice. In percolation models 1 and 2, losses can be unheralded, and $\eta > \lambda_\eta$ thus represents only a necessary condition, as discussed before.

IV. UNION-FIND ALGORITHM FOR SIMULATING PHOTON LOSS

The percolation models introduced above give the loss tolerance of fusion networks or graph states as parameterized by the percolation threshold λ_η. In this section, we give efficient algorithms for computing λ_η. For all percolation models, it is not obvious which lattice is the best choice [14]. In classical bond or site percolation, a larger vertex degree (coordination number) of the lattice is typically helpful to lower the percolation threshold. In contrast, when a loss is compensated by removing all neighbors of a lost photon by Z-basis measurements [see Fig. 1(b)], a graph or fusion network with a high vertex degree is particularly sensitive to photon loss. However, a too-low vertex degree makes the graph state or fusion network fragile due to the higher bond-percolation threshold [10,11]. To find the optimum in between and simulate various fusion lattices, a fast algorithm for computing percolation thresholds λ_η is required. This algorithm should ideally scale linearly with the number of qubits/photons and ideally should be independent of the number n_η of sampled values for η. In the following, we first describe the well-known Newman-Ziff algorithm [23], which applies to bond or site percolation. In the subsequent sections, we describe modified algorithms that can be applied to the percolation models 1 to 3.[7]

The easiest way to perform classical bond- or site-percolation simulations is in the canonical ensemble where the bond (site) probabilities p_{bond} (p_{site}) are fixed and bonds/sites are randomly occupied according to these probabilities. Connected graph components, their size, or whether they percolate the lattice can then be obtained with a time complexity that is linear in the system size by depth- or breadth-first graph traversal [59]. Constructing a lattice with constant vertex degree (coordination number z), a number of nodes $|V|$, and a number of edges $|E| = \frac{z}{2}|V|$ takes $\mathcal{O}(|V|)$ and traversing the graph (e.g., breadth first) takes $\mathcal{O}(|V|)$ as well [23]. In some two-dimensional percolation simulations, this scaling

---

[7]Similar modifications of the Newman-Ziff algorithm have been developed for other classical percolation models such as bootstrap or diffusion percolation [58].
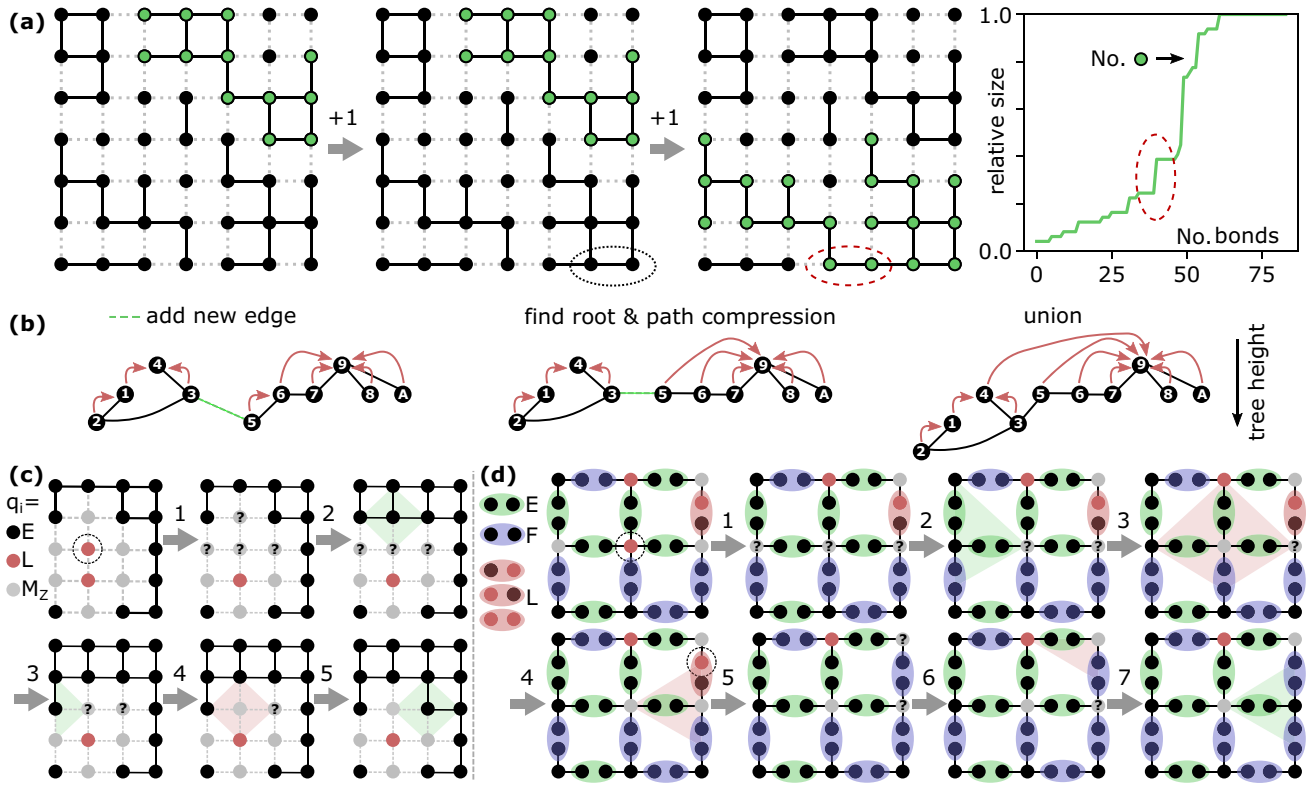
FIG. 2. Algorithms to speed up percolation simulations. (a) The key idea of the Newman-Ziff algorithm [23] illustrated for bond percolation. Bonds are added one by one in random order and the largest connected component of the graph (green nodes) is monitored. When two large clusters merge, there is a rapid increase in the size of the largest connected component, as shown on the right. (b) Updating the connected graph components in classical bond percolation [23]. In the first step (find), the root nodes of the two components that the new edge connects are identified by following the tree data structure (red arrows). In the second step (union), the two components are merged by connecting the root node of the smaller component to the root node of the larger one. (c) Illustration of Algorithm 1 for simulating the effect of loss on a graph state (percolation model 1). Initially, two qubits are lost (red nodes) and the qubits in their neighborhood are therefore measured in the Z basis (gray). When adding one missing qubit in step 1, several qubits in the neighborhood potentially need to be updated. This update is done in steps 2–5, where the neighborhoods of all potentially affected nodes are checked for lost qubits. When none of a qubit's neighbors are lost, there is no longer any need to measure it in the Z basis and it is added to the graph. (d) Illustration of Algorithms 2a and 2b for simulating the effect of loss when creating a large graph state by fusing star-shaped resource states (percolation model 2). Algorithm 2a: In step 1, a lost central qubit is added (encircled node). All qubits that are not lost and connected to the added qubit by a successful fusion potentially need to be updated. The update is done in steps 2–4 by applying the rules illustrated in Fig. 1 to all potentially affected qubits. Algorithm 2b: After step 4 is completed, we assume in step 5 that a previously lost fusion photon is added (encircled node). The subsequent update algorithm (steps 6 and 7) is similar to before with the difference that only the central node belonging to the added fusion photon and the central node belonging to its fusion partner need to be updated.

can be slightly improved by parsing only the boundary of a spanning cluster [23,60]. However, this method does not solve the general issue described in the following.

To make a percolation simulation for n, different probabilities of bond occupation (p_{bond}) or site occupation (p_{site}), the computation has to be repeated n times and the running time becomes $\mathcal{O}(n|V|)$ [23]. When many different probabilities p_{bond} or p_{site} are considered, the factor n becomes a major hurdle. For classical bond- and site-percolation simulations, an algorithm that avoids this issue by considering the problem in the microcanonical ensemble has been developed by Newman and Ziff [23]. In this algorithm, the number of occupied bonds/sites is fixed (not the bond/site probability as before). In the corresponding site-percolation simulations, all nodes are initially removed and added to the graph in random order,

one by one. [Bond percolation can be performed analogously by adding bonds as illustrated in Fig. 2(a).] Connected graph components of existing nodes are stored and when a new node is added, the corresponding data structure storing the connected components is updated. At the same time, one keeps track of the size of the largest connected component $S(i)$ and a list of Booleans $B(i)$ that specifies if percolation (cluster spanning [23]) has been achieved after adding node number i. Having computed $\langle S(i) \rangle$ by averaging several simulations of $S(i)$, the expectation value of the largest component size can be computed as a function of p_{site} by [8]

$$\langle S(p_{site}) \rangle = \sum_{i=0}^{|V|} \langle S(i) \rangle \binom{|V|}{i} p_{site}^i (1 - p_{site})^{|V|-i}, \quad (3)$$

where the binomial coefficients are computed in a normalized way using the iterative method from Ref. [23] (for large $|V|$, a Gaussian approximation of the binomial distribution [61] is an alternative). Replacing $S(i)$ by $B(i)$ in the above equation computes the probability of percolation for a given probability p_{site}. It has been shown that the above percolation simulation has a time complexity of only $\mathcal{O}(|V|)$ (with no factor n) if a suitable union-find algorithm (see below) is used for merging graph components [23,62].

Critical to implementing such an algorithm is a data structure that efficiently keeps track of the graph and its connected components, as well as a method to update it when adding an edge/node. As in Ref. [23], we use a tree data structure [63] where every node is either a root node representing an isolated graph component or it points to a parent node that is part of the same graph component. An example of such a data structure and how to update it for bond percolation is shown in Fig. 2(b), where the simulated graph is drawn in black and the tree data structure is illustrated by the red arrows. Adding a previously missing edge is done by updating the graph and the tree data structure using the union-find algorithm from Ref. [23]: Following the tree data structure, the algorithm first determines the root nodes of the two graph components that the new edge connects (find). (To reduce the length of subsequently traversed paths, we employ path compression [23,62] such that the parent node for all nodes along the path is updated to the root node.) If the two root nodes are different, the graph components are merged by making one of the root nodes point to the other (union). To improve the performance, one attaches the smaller component to the larger one (weighted union [23,62]). This union operation is done in $\mathcal{O}(1)$ steps as opposed to an approach based on an array (lookup table) storing the root node for every node. The latter would need $\mathcal{O}(1)$ for a single find operation, but $\mathcal{O}(|V|)$ to determine all nodes of the cluster that must be updated in a union. The overall time complexity would become $\mathcal{O}(|V|^2)$ in the worst case. In contrast, the combination of path compression and weighted union using the tree data structure results in a practically linear time scaling, $\mathcal{O}(|V|)$.[9] This result is important for efficiently performing classical bond- or site-percolation simulations [23] and also is applied in union-find decoders for quantum error correction [64].

Adding a previously missing node in a site percolation simulation is done similarly [23]: The algorithm first adds the new node as an isolated single-node graph component. Then it adds the edge to the first neighbor and updates the graph, as in

---

[8]For most practical cases, the computation of $S(i)$ is the most time-consuming part of the simulation, although calculating Eq. (3), technically, has a worse scaling behavior. For large graphs, however, it is only necessary to take $\mathcal{O}(\log |V|)$ elements of the sum because of the rapid decrease of the binomial distribution away from its maximum. So the running time for doing this summation for n different values of p_{site} becomes $\mathcal{O}(n \log |V|)$.

[9]There is a nonconstant prefactor which, however, is not expected to exceed 3 in a system of any realistic size (see Refs. [23,62] for the discussion of the related Ackermann's function).

the case of bond percolation.[10] This procedure is repeated for all the other neighbors. We refer to this algorithm for adding a previously missing node as Algorithm 0 (see Ref. [23] for more details).

ALGORITHM 0: Add new node.

---

1 Add a new node to the graph and update the tree data structure with the algorithm from Ref. [23].

In the following, we describe algorithms that use the tools described so far to efficiently simulate the percolation models 1 to 3 and obtain the corresponding percolation threshold λ_η. Algorithm 1 applies to the loss tolerance of a given cluster state (percolation model 1). Algorithms 2a and 2b apply to fusion networks (percolation model 2). Algorithm 2a applies to lost central qubits of the star-shaped resource states and thus only is used when the central qubits are photons that can be lost (percolation model 2). Algorithm 2b applies to fusion photons and thus is used independently of whether the central node is a photon (percolation model 2) or a quantum emitter (percolation model 2'). Finally, Algorithm 3 applies to a repeat-until-success fusion network (percolation model 3).

ALGORITHM 1: Add lost qubit ($q_i = L$) to the graph state.

```
   /* step 1:  change label of lost qubit       */
1  q_i := M_Z;
   /* step 2:  list potentially affected qubits  */
2  Ṽ := {j ∈ N(i) | q_j = M_Z} ∪ {i};
   /* step 3:  update and add qubits              */
3  for j ∈ Ṽ do
4  |    if (∄ k ∈ N(j) : q_k = L) then
5  |    |    q_j := E;
6  |    |    add node j via Algorithm 0;
```

ALGORITHM 2a: Add lost central qubit ($q_i = L$) to the fusion network.

```
   /* step 1:  change label of lost qubit       */
1  q_i := M_Z;
   /* step 2:  list potentially affected qubits  */
2  Ṽ := {j ∈ N(i) | q_j = M_Z ∧ e_{ij} = E} ∪ {i};
   /* step 3:  update and add qubits              */
3  for j ∈ Ṽ do
4  |    if (∄ k ∈ N(j) : q_k = L ∧ e_{jk} = E) ∧
5  |    (∄ k ∈ N(j) : e_{jk} = L) then
6  |    |    q_j := E;
7  |    |    add node j via Algorithm 0;
```

---

[10]Note that some edges might be labeled as missing due to a finite bond probability (or, for some of the following algorithms, a finite fusion probability). In this case, only existing edges are considered.

ALGORITHM 2b: Add lost fusion photon on edge e_{ij} of fusion network.

```
    /* step 1:  check if any update is needed    */
1   c_{ij} := c_{ij} + 1;
2   if c_{ij} = 2 then
3   |   with probability p_s: e_{ij} := E;
4   |   otherwise: e_{ij} := F;
5   else
6   |   skip steps 2, 3;
    /* step 2:  list potentially affected qubits  */
7   Ṽ := {k ∈ {i, j} | q_k ≠ L};
    /* step 3:  update and add qubits             */
8   for l ∈ Ṽ do
9   |   if (∄ k ∈ N(l) : q_k = L ∧  e_{lk} = E) ∧
10  |      (∄ k ∈ N(l) : e_{lk} = L) then
11  |   |   q_l := E;
12  |   |   add node l via Algorithm 0;
```

ALGORITHM 3: Add previously lost fusion photon on edge e_{ij} of adaptive fusion network.

```
    /* step 1:  check if any update is needed    */
1   c_{ij} := c_{ij} + 1;
2   if c_{ij} = 2 · n_{ij} then
3   |   if b_{ij} = 1 then
4   |   |   e_{ij} = F;
5   |   else
6   |   |   e_{ij} = E;
7   else
8   |   skip all following steps;
    /* step 2:  list potentially affected qubits  */
9   Ṽ := {i, j};
    /* step 3:  update and add qubits             */
10  for l ∈ Ṽ do
11  |   if (∄ k ∈ N(l) : e_{kl} = L) then
12  |   |   q_l := E;
13  |   |   add node l via Algorithm 0;
```

A. Update rule for percolation model 1

First, we consider percolation model 1 where an existing photonic cluster state suffers loss. Similar to the Newman-Ziff algorithm [23], we initially assume that all photons are lost and then add them one by one, keeping track of the largest connected graph component or checking if a graph component percolates. However, new rules are required to update the graph when a previously lost photon is added. Updating the cluster state after adding a previously lost qubit i is complicated by the fact that the neighborhood $N(i)$ of a lost qubit used to be measured in the Z basis [5]. When adding a missing qubit, those neighbors potentially have to be updated and added to the graph if there is no longer any need to measure them in the Z basis. We assign a label q_i to every central qubit of a resource state: $q_i = L$, meaning that the qubit is lost; $q_i = M_Z$, meaning that the qubit needs to be measured in the Z basis; or $q_i = E$, meaning that the qubit exists (neither lost nor has to be measured in the Z basis, and therefore is part of the

graph/cluster state). Algorithm 1 describes the corresponding data structure update when changing the label of a qubit in a cluster state from lost to not lost (decreasing the number of lost photons by one).

The algorithm is illustrated in Fig. 2(c). In the first step, the label of the lost qubit i is changed to $q_i = M_Z$, indicating that it is no longer lost. In the second step, a list $\tilde{V}$ of all potentially affected qubits is created [all qubits in the neighborhood $N(i)$ of qubit i as well as qubit i itself]. In the final step, the neighborhood of all qubits in $\tilde{V}$ is checked for lost qubits. If a lost qubit is still found in the neighborhood of a qubit $j \in \tilde{V}$, no update needs to be done. If no lost qubit is found in the neighborhood $N(j)$, the label of the qubit is updated to E. In the latter case, qubit j was only affected by the loss of qubit i and can be added to the graph once qubit i is no longer lost. Adding qubits to the graph is done with Algorithm 0 from Ref. [23].

B. Update rule for percolation models 2 and 2′

Experimentally more realistic than percolation model 1 is the construction of a large graph state by fusing many small star-shaped resource states (percolation model 2) [5,14]. In this percolation model, fusions between *leaf* nodes are used to establish connections and every photon (either in the center of a star-shaped resource state or a fusion photon on the *leaf*) can be lost with a uniform probability $p_{\text{loss}} = 1 - \eta$. We assign different labels to every edge e_{ij} (i, j are the numbers of the central qubits of the star-shaped resource states that the edge connects upon successful fusion): $e_{ij} = E$ when the corresponding fusion was successful, $e_{ij} = F$ when the fusion failed but no photon was lost, and $e_{ij} = L$ when at least one of the photons on the edge is lost. In the case of loss, we also need to keep track of whether one ($c_{ij} = 1$) or zero ($c_{ij} = 0$) fusion photons on edge e_{ij} are present (not lost), i.e., whether both or just one photon was lost.

When adding a previously lost photon, there are two cases: the photon can be a central qubit of the star-shaped resource state or it can be a fusion photon on a leaf node. We first consider the case of adding a lost central qubit to the corresponding fusion network (applies to percolation model 2, but not 2′). The graph update is done by Algorithm 2a [see Fig. 2(d) for an illustration]. Here the neighborhood $N(i)$ refers to all qubits in the center of other star-shaped resource states that one tries to connect to qubit i by a fusion. In the first step, the label of the lost qubit is changed to $q_i := M_Z$ as in Algorithm 1. In the second step, a list $\tilde{V}$ of all potentially affected qubits is created. A qubit $j \neq i$ can only be affected by a change of qubit i when there is a connection between both qubits obtained by a successful fusion ($e_{ij} = E$). In the final step, all potentially affected qubits $j \in \tilde{V}$ are updated. A qubit can only be added ($q_j := E$) when it is not connected to a lost qubit by a successful fusion [$\nexists k \in N(j) : q_k = L \wedge e_{jk} = E$] and no fusion photon on any of its edges is lost [$\nexists k \in N(j) : e_{jk} = L$].

Next, we consider the case where the lost photon is a fusion photon between two central qubits i, j. This case applies to both percolation model 2 and percolation model 2′, with an example shown in the lower part of Fig. 2(d). Here, Algorithm 2a needs to be slightly modified: First, the edge e_{ij} must be

updated if and only if the fusion partner of the new qubit is not lost ($c_{ij} = 1$ before the update), otherwise $e_{ij} = L$ remains unchanged and nothing else needs to be done. In the former case, the edge label is updated to $e_{ij} = F$ or $e_{ij} = E$ with probabilities of $1 - p_s = p_s = 1/2$, and steps 2 and 3 are performed. Second, the list of affected central qubits in step 2 must be changed to $\tilde{V} = \{i, j\}$. Algorithm 2b describes the corresponding graph update rule.

C. Update rule for percolation model 3

So far, we have explored how the Newman-Ziff algorithm can be modified to simulate the effect of photon loss in fusion networks where every fusion is performed once, independent of the outcome of other fusions. In percolation model 3, the outcome of previous fusions determines whether or not a fusion is repeated and the algorithms developed so far thus cannot be applied. In Algorithm 2a, for instance, we initially labeled all photons as lost (L) and added them one by one. In percolation model 3, the overall number of photons is not constant and it is therefore unclear how many photons exist and where in the graph they are used. We show how a modification of the Newman-Ziff algorithm nevertheless can be applied to the adaptive repeat-until-success fusion network described by percolation model 3.

In percolation model 3, a fusion is repeated until it either succeeds, fails after a maximum number of repetitions, n_{max}, or heralds photon loss before completion. An exemplary tree diagram for the different fusion outcomes and the associated probabilities is given in Fig. 3(a). The probabilities that the fusion attempts on a certain edge succeed ($p_{\tilde{S}}$), fail ($p_{\tilde{F}}$), or terminate with a loss ($p_{\tilde{L}}$) are [54,65]

$$p_{\tilde{F}} = [\eta^2(1 - p_s)]^{n_{max}} = \alpha^{n_{max}}, \tag{4}$$

$$p_{\tilde{S}} = \eta^2 p_s \sum_{k=0}^{n_{max}-1} [\eta^2(1 - p_s)]^k = \alpha \frac{\alpha^{n_{max}} - 1}{\alpha - 1}, \tag{5}$$

$$p_{\tilde{L}} = 1 - p_{\tilde{F}} - p_{\tilde{S}}, \tag{6}$$

where we have used $p_s = 0.5$ for the success rate of a single fusion attempt and defined $\alpha \equiv \eta^2/2$. In the case of photon loss, the two quantum emitter qubits are removed from the graph by Z-basis measurements, as before.[11]

Our algorithm to simulate percolation model 3 consists of two main steps. In the first step, we assume that there is no photon loss and determine, for every edge e_{ij}, how many fusion attempts are required for the fusion to succeed. We label this number of fusion attempts n_{ij} and it is determined by tossing a random coin until the coin indicates success or we reach the maximum number of allowed fusions, n_{max}. The probability distribution for the frequency of n_{ij} thus decays exponentially for large values of n_{ij}. When none of the n_{max} fusions succeed, we keep track of this by a Boolean array b_{ij}, where $b_{ij} = 0$ indicates fusion success and $b_{ij} = 1$ fusion
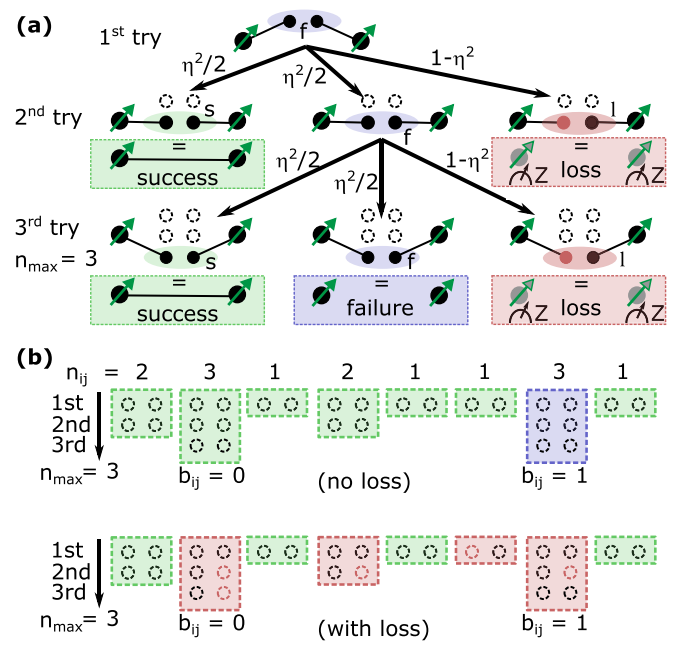

FIG. 3. (a) Adaptive scheme where fusions are repeated until either success, heralded photon loss, or $n_{max} = 3$ trials is reached. We have assumed here that the first fusion attempt fails. The scheme can be realized using quantum emitters with a spin (drawn as black dots with green arrows) and proceeds by adjacent quantum emitters generating new fusion photons if an attempt to entangle them with a fusion fails. When one or more fusion photons are lost, both quantum emitter spins need to be reinitialized or measured in the Z basis. For every single fusion attempt, the corresponding outcome is indicated by the color of the ellipses that indicates the fusion as well as a letter: s and green ellipse for fusion success (probability $\eta^2 p_s = \eta^2/2$), f and blue ellipse for fusion failure [probability $\eta^2(1 - p_s) = \eta^2/2$], or l and red ellipse for a fusion with a loss (probability $1 - \eta^2$). The dashed rectangular boxes indicate the final outcome of the repeat-until-success fusion when following the corresponding path. (b) Schematic illustration of the data structure used for keeping track of the fusion results when sweeping the number of lost photons. Every dashed box represents a fusion that is repeated in case it does not succeed. The color represents the final outcome of the fusion (green: success; blue: failure; red: loss) and every dashed circle inside the boxes represents a fusion photon, with red color indicating that the photon is lost. In the absence of loss (upper part), the overall fusion succeeds, except when all individual fusion attempts fail ($b_{ij} = 1$). As long as there is one lost photon in the box representing the overall sequence of fusion attempts ($c_{ij} < 2n_{ij}$), the fusion terminates by re-initializing the spin qubit.

failure. The number n_{ij} determines how many potentially lost photons we have to consider for the edge e_{ij}, namely, exactly $2n_{ij}$, as illustrated in Fig. 3(b). The reason is that after n_{ij} fusion attempts, no further fusions on e_{ij} are performed because the fusion either succeeded, or failed when $n_{ij} = n_{max} \wedge b_{ij} = 1$, or because there has been a heralded photon loss before. So without loss, the overall number of fusion photons of all fusion attempts is $\sum_{ij} 2n_{ij}$.

In the second step, the fusion photons of all the edges are labeled as lost and are added, one by one, in random order while updating the connected graph components. In that

---

[11]This step is not required for systems where entangler and memory qubit are two different systems that can be coupled/decoupled on demand [30,56,66]. For quantum emitters such as quantum dots, such techniques are, however, only at a very early stage [67].

process, an edge keeps the label $e_{ij} = L$ until all its $2n_{ij}$ fusion photons are no longer lost. We count, for every edge e_{ij}, the number of photons that are not lost with a counter variable c_{ij} (initially zero and increasing when a fusion photon is added on edge e_{ij}). A change of the graph needs to be done once c_{ij} reaches $2n_{ij}$ (before that, the attempt to establish a connection between nodes i and j always terminates with a photon loss).[12] Algorithm 3 describes how this update is done. As the central qubits q_i are quantum emitters (and thus cannot be lost), Algorithm 3 differs from Algorithm 2b in steps 2 and 3.

Finally, we note that an alternative method for boosting the fusion success probability uses ancilla entangled photon pairs [47] and this approach could be similarly modeled. The probabilities for fusion success and loss of a fusion photon in this case differ from Eqs. (5) and (6) as 2^n photons are used for every fusion ($n = 1$ being the standard fusion) to achieve a success probability of $1 - 1/2^n$. We, therefore, would have $p_{\bar{F}} = \eta^{(2^n)}/2^n$, $p_{\bar{S}} = \eta^{(2^n)}(1 - 1/2^n)$, $p_{\bar{L}} = 1 - \eta^{(2^n)}$. Using 2^n photons per fusion can be simulated with the approach from before by setting $2n_{ij} = 2^n$. Once all fusion photons exist, the fusion is set to failure with a probability of $1/2^n$ and success otherwise. Alternatively, since the number of photons is fixed for the all-photonic boosting, it would be possible to make a modified Newman-Ziff algorithm by adding lost fusions one by one instead of virtually adding all individual lost photons. The size of the largest connected graph component is then obtained as a function of the number of fusions with no lost photons. Using the relation between η and $p_{\bar{L}}$, one can convert such a simulation into the largest connected graph component as a function of η by rescaling the x axis. For percolation model 3, such a simplification is, however, not possible because $p_{\bar{S}}/p_{\bar{F}}$ is not constant as a function of η. Even if one would compute $p_{\bar{S}}/p_{\bar{F}}$ (by converting $p_{\bar{L}}$ into η) in every step where a previously lossy fusion is added, all the fusions that were added before would appear with a ratio $p_{\bar{S}}/p_{\bar{F}}$ that is inconsistent with the current fraction of nonlossy fusions (respectively, $p_{\bar{L}}$). Also, when fusion photons as well as other photons are lost (Algorithm 2a), the only feasible approach is virtually adding photons one by one and doing the required data structure update in every step.

V. RESULTS AND RUNNING TIME

We have implemented the described algorithms for the various percolation models and the source code can be found in Ref. [31]. More details on the source code and how we verify parts of the implementation can also be found in Appendix.

In this section, we use these implementations for several percolation simulations. Figure 4(a) shows an exemplary per-

colation simulation (percolation model 2') of the simple cubic lattice for different lattice sizes. From such a simulation, we can obtain the corresponding percolation threshold λ_η. Figure 4(b) shows the obtained percolation thresholds for the percolation models 1, 2, and 2' on several different lattices. More simulation results for percolation models 2 and 2' can be found in Ref. [14].

Figure 4(c) shows the percolation thresholds for percolation model 3 using the hypercubic (hc) fusion lattices [68]. These simulations are performed for different values of the maximum number of fusion repetitions ($n_{\max}$). We observe that the adaptive scheme investigated here improves the loss tolerance for the low-dimensional lattices, but makes it worse in higher dimensions. This trend is also observed for other fusion lattices that we investigated. The reason for the worse loss tolerance is that repeating the fusions until success increases the chance that the fusion terminates by photon loss. Using all-photonic boosting [47,48] would have a similar issue. Of course, the illustrated adaptive scheme is not ideal: fusions are, for instance, performed even if it is known that one of the involved quantum-emitter qubits must be measured in the Z basis due to a loss in a previous fusion. Problems such as this can be avoided with more sophisticated adaptive schemes.

Next, we consider the running times of the percolation simulations. When using Algorithms 1, 2a, 2b, or 3, we expect that the running time is independent of the number n_η of simulated photon loss probabilities. Without these algorithms, a new graph traversal needs to be performed for every efficiency $\eta = 1 - p_{\text{loss}}$ leading to a time that is linear in n_η. Figure 4(d) shows the measured running times as a function of n_η, confirming the expected time scalings with n_η. We find that the simulations using the modified Newman-Ziff algorithms become more efficient when about a dozen different values for η are simulated. The speed advantage is key for determining the loss tolerance of large fusion networks [14].

Furthermore, we expect that the running time as a function of the system size (number of vertices $|V|$ in the cluster state or fusion network) scales identically to Ref. [23], meaning practically linear with $|V|$. The reason is that the overhead of Algorithms 1 and 2a compared to Algorithm 0 is a constant prefactor that depends on the vertex degree of the simulated lattice. This intuition is supported by Fig. 4(e), where we find an approximately linear scaling of the running time when increasing the size of the simulated lattice.

VI. SUMMARY AND OUTLOOK

We have shown that it is possible to use a modification of Newman and Ziff's algorithm [23] for nonstandard percolation models motivated by photon loss in cluster states and fusion networks. We have considered ballistic fusion networks where many resource states are connected by simultaneous probabilistic fusions and an adaptive scheme where fusions are repeated until success. The developed algorithms speed up the simulation of the considered architectures, which enables running optimizations over many fusion lattices [14].

---

[12]This algorithm works because the following is fulfilled: assume that a fraction $\tilde{\eta} = \sum_{ij} c_{ij} / \sum_{ij} 2n_{ij}$ of all fusion photons has been updated to not lost. Then the probabilities that the result of the fusion attempts on a random edge e_{ij} is labeled as failure, success, or loss are equal to the actual probabilities $p_{\bar{F}}, p_{\bar{S}}, p_{\bar{F}}$ in Eqs. (4), (5), and (6). Thus, Eq. (3) (with $|V|$ being replaced by the number of all photons, $\sum_{ij} 2n_{ij}$, and the averaging being done outside the sum) yields the expectation value of the largest graph component size as a function of $\tilde{\eta}$.

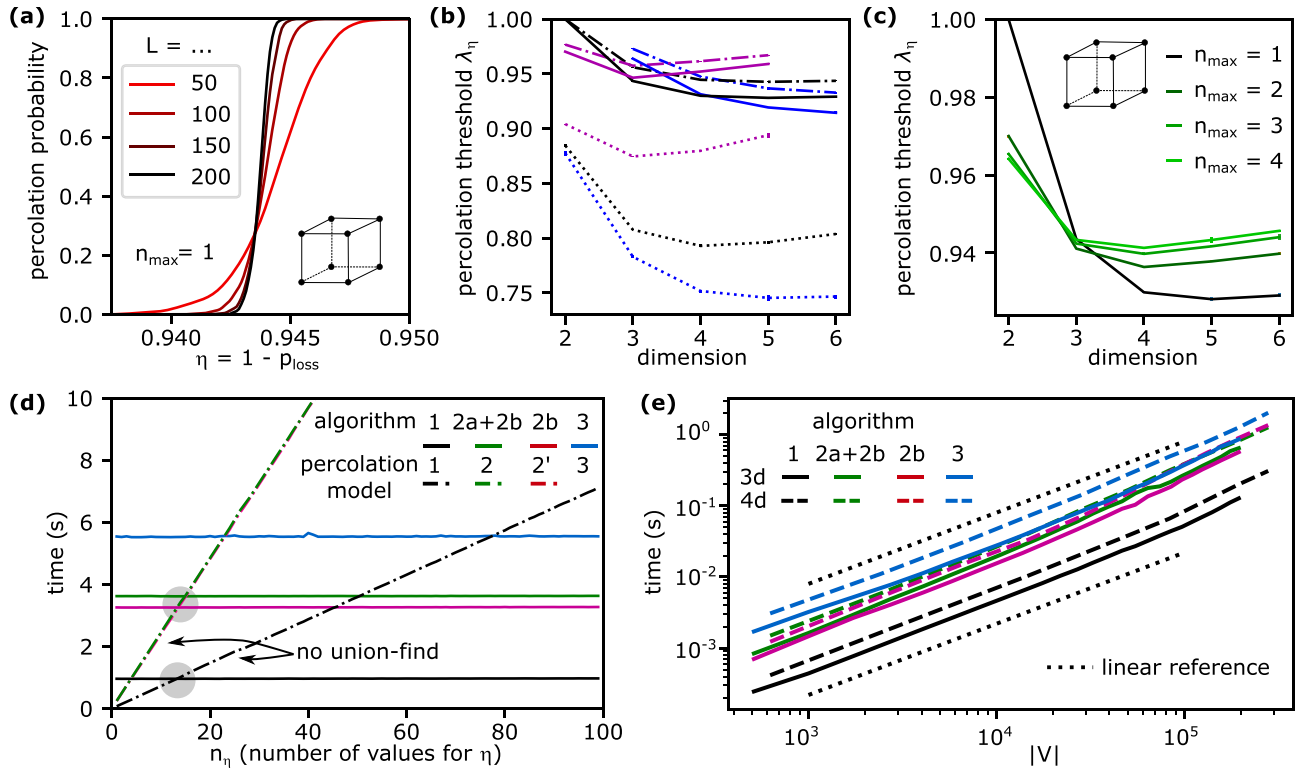[13]Used system: Rocky Linux 8.7 (Green Obsidian), AMD Opteron(TM) Processor 6276 (single core used per simulation).

FIG. 4. (a) Simulation of the simple cubic fusion lattice for percolation model 2′ (equivalent to percolation model 3 with $n_{max} = 1$). The size of the simulated lattices is specified by the parameter L, which is the number of unit cells along all three dimensions. (b) Percolation threshold λ_η for the diamond (blue), hypercubic (black), and bcc+hc lattices (purple). The dotted lines show the results for percolation model 1, the dashed lines are simulations of all-photonic fusion networks (percolation model 2), and the solid lines correspond to simulations with quantum emitters as the central qubits (percolation model 2′). (c) Percolation threshold of the hypercubic fusion lattice for different values of n_{max} (percolation model 3). (d) Running time for simulating a three-dimensional (3D) simple cubic lattice with $L = 100$ and so $L^3 = 10^6$ central nodes, averaged over 20 repetitions.[13] The simulations are performed for n_η different values of $\eta = 1 - p_{loss}$. When using Algorithms 1, 2a, 2b, and 3, the simulation time is constant, independent of n_η (solid lines). Without these algorithms, the simulation time increases linearly with n_η (dash-dotted lines). The gray circles indicate the number n_η where the modified Newman-Ziff algorithms become more efficient. (For percolation model 3, we have only implemented a modification of the Newman-Ziff algorithm and thus no dash-dotted line is shown.) (e) Running time of different percolation simulations as a function of the lattice size $|V|$. In the case of fusion networks (percolation models 2, 2′, 3), $|V|$ represents the number of central qubits, not counting the fusion photons. The running time of a full percolation simulation (averaged over 300 repetitions) scales linearly with $|V|$ for a three-dimensional (solid lines) and a four-dimensional (dashed lines) simple cubic lattice. The dotted lines indicate linear scaling for reference.

Mitigating the photon loss with such optimizations is highly important as loss is arguably the biggest challenge for photonic quantum computing and networking.

We have considered star-shaped resource states, but our algorithms also apply when resource states with a graph structure of a caterpillar tree are used (like star-shaped resource states, these states can be generated by a single quantum emitter [6]). The only condition for our algorithms to work is that all fusion photons are *leaf* nodes of the resource state graphs and thus only have a single neighbor, which also may apply to some graph states that cannot be generated by a single quantum emitter [69,70]. Whether similar algorithms can be applied when fusing other types of states, such as linear chain graph states [24,35], could be investigated in future works. Furthermore, policies other than repeat-until-success could be used in the adaptive scheme to decide whether or not a fusion is applied. It would be interesting to investigate if similar modifications of the Newman-Ziff algorithm could be applied to such percolation models.

Finally, we hope that our results might be inspiring or useful for other applications in quantum information: percolation models can be applied to quantum networks [71], the bond-percolation threshold of the syndrome graph determines the loss threshold of various topological quantum error-correction codes [7,34,72–74], and qubit loss in color codes can be simulated with modified percolation models [75,76]. In addition, logical errors in topological codes are operators that span around the code [77]. The transition where errors plus the decoder output give a percolating operator is thus a lower bound for the error threshold [78]. All these points indicate that similar algorithms as the ones developed here could improve the analysis of quantum error correction in general.

All data and source code from this work can be found in Ref. [79]. An updated version of the source code can be found in Ref. [31]. More data, shown in Ref. [14], can be found in Ref. [80].

APPENDIX: SOURCE CODE

1. General description

Our simulation program is written in the programming language C and is provided as an open-source repository [31]. It can be used to simulate λ_η for percolation models 1, 2, 2′, and 3 with various lattice geometries in several dimensions. Furthermore, it enables simulating standard bond or site percolation using the Newman-Ziff algorithm. It also applies to site-bond or bond-site [21] percolation simulations (where one parameter being either the bond or site probability is fixed and the other parameter is simulated over a continuous range using the Newman-Ziff algorithm). The following explanation focuses on percolation models 2 and 2′, but the other models are implemented very similarly.

Every percolation simulation consists of three main steps: In the first step, a certain lattice with a fixed size is constructed. We use an adjacency list to represent the graphs/fusion lattices, keeping the data structure analogous to Ref. [23]. Therefore, other lattices can be easily built or taken from the literature where a similar implementation is used [61]. In the second step, the effect of the random process of photon loss and fusion failure is simulated. Since the fusion probability is a fixed parameter given by the physical implementation, we first randomly set fusions to success/failure. The effect of photon loss is then simulated using the described algorithms where all photons are labeled as lost initially and then added one by one. During that process, we keep track of the largest component size $S(i)$ or a Boolean array $B(i)$ specifying if percolation (cluster spanning) has been achieved, with i specifying the number of photons that are not lost. This second part of the simulation is repeated several times for averaging. We estimate the percolation threshold for a given lattice size as the fraction of nonlost photons η where cluster spanning is achieved on average when adding photons one by one.[14] In the third step, we determine the percolation threshold in the limit of infinite lattice size by compensating

_____

[14]Several different methods could be used instead to determine the percolation threshold [81–83].

for finite-size effects with an extrapolation (see Ref. [68]). Furthermore, to obtain the expectation value for the probability of percolation or the largest connected component size as a function of the parameter η, we use Eq. (3). In this way, a plot such as, for instance, the one shown in Fig. 4(a) is obtained.

For the graph construction in the first step, we provide several functions that construct lattices such as hypercubic, diamond, bcc, and fcc in a dimension of choice. In particular, our implementation allows simulating all lattices from Ref. [84] including their higher-dimensional generalizations, all lattices from Ref. [68] except the kagome lattice, and hypercubic lattices with extended neighborhoods [85–87].

2. Verification of the implementation

To verify our lattice constructions, we have performed classical site-percolation simulations of lattices with known percolation thresholds [68,84]. These simulations can be found in Ref. [14], where we have simulated two- to six-dimensional lattices that for periodic boundaries, correspond to various k-regular graphs (meaning that every node has exactly k neighbors) with $k \in [3 \ldots 60]$. Additionally, we compute here a few bond-percolation thresholds. Note that none of these simulations represent a central part of this work, yet an agreement with the literature values is a useful consistency check to verify that the lattices are correctly defined and implemented.

For the bond-percolation thresholds of the generalized diamond lattices [68], we find $\lambda_{\text{bond}} = 0.6529(10)$, $0.3893(7)$, $0.2709(18)$, $0.2072(15)$, $0.1646(43)$ for dimensions 2–6, respectively, which is in agreement with Refs. [68,88,89]. Furthermore, we find $\lambda_{\text{bond}} = 0.2491 \pm 0.0002$ for the bond-percolation threshold of the simple cubic lattice (that roughly agrees with Ref. [90]), which also represents the syndrome graph of the Raussendorf-Harrington-Goyal (RHG) lattice [57]. For the latter, we find a bond-percolation threshold of $\lambda_{\text{bond}} = 0.3845(1)$ and a site-percolation threshold of $\lambda_{\text{site}} = 0.4220(5)$.

The agreement of several simulations with the literature values verifies that our implementation of the corresponding lattices is correct. However, the studied percolation models for photon loss as well as the Algorithms 1, 2a, 2b, and 3 have not been considered in the literature. The bond- and site-percolation simulations, therefore, do not say anything about the correctness or the correct implementation of these algorithms. To verify that the implementation of our algorithms is correct, we have implemented several redundant functions for the corresponding percolation models without any Newman-Ziff-type algorithm (see the readme file of the code repository [31] for more information). We have performed several tests for which the results were consistent between the different implementations.

[1] M. Hein, W. Dür, J. Eisert, R. Raussendorf, M. Nest, and H.-J. Briegel, Entanglement in graph states and its applications, arXiv:quant-ph/0602096.

[2] R. Raussendorf and H. J. Briegel, A one-way quantum computer, Phys. Rev. Lett. **86**, 5188 (2001).

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

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

[5] M. Gimeno-Segovia, P. Shadbolt, D. E. Browne, and T. Rudolph, From three-photon Greenberger-Horne-Zeilinger states to ballistic universal quantum computation, Phys. Rev. Lett. **115**, 020502 (2015).

[6] S. Paesani and B. J. Brown, High-threshold quantum computing by fusing one-dimensional cluster states, Phys. Rev. Lett. **131**, 120603 (2023).

[7] S. Bartolucci, P. Birchall, H. Bombin, H. Cable, C. Dawson, M. Gimeno-Segovia, E. Johnston, K. Kieling, N. Nickerson, M. Pant *et al.*, Fusion-based quantum computation, Nat. Commun. **14**, 912 (2023).

[8] S. Morley-Short, S. Bartolucci, M. Gimeno-Segovia, P. Shadbolt, H. Cable, and T. Rudolph, Physical-depth architectural requirements for generating universal photonic cluster states, Quantum Sci. Technol. **3**, 015005 (2018).

[9] D. Herr, A. Paler, S. J. Devitt, and F. Nori, A local and scalable lattice renormalization method for ballistic quantum computation, npj Quantum Inf. **4**, 27 (2018).

[10] K. Kieling, T. Rudolph, and J. Eisert, Percolation, renormalization, and quantum computing with nondeterministic gates, Phys. Rev. Lett. **99**, 130501 (2007).

[11] M. Pant, D. Towsley, D. Englund, and S. Guha, Percolation thresholds for photonic quantum computing, Nat. Commun. **10**, 1070 (2019).

[12] E. Knill, R. Laflamme, and G. J. Milburn, A scheme for efficient quantum computation with linear optics, Nature (London) **409**, 46 (2001).

[13] B. Hacker, S. Welte, G. Rempe, and S. Ritter, A photon-photon quantum gate based on a single atom in an optical resonator, Nature (London) **536**, 193 (2016).

[14] M. C. Löbl, S. Paesani, and A. S. Sørensen, Loss-tolerant architecture for quantum computing with quantum emitters, Quantum **8**, 1302 (2024).

[15] A. Chanana, H. Larocque, R. Moreira, J. Carolan, B. Guha, E. G. Melo, V. Anant, J. Song, D. Englund, D. J. Blumenthal *et al.*, Ultra-low loss quantum photonic circuits integrated with single quantum emitters, Nat. Commun. **13**, 7693 (2022).

[16] R. Uppu, L. Midolo, X. Zhou, J. Carolan, and P. Lodahl, Quantum-dot-based deterministic photon-emitter interfaces for scalable photonic quantum technology, Nat. Nanotechnol. **16**, 1308 (2021).

[17] N. Tomm, A. Javadi, N. O. Antoniadis, D. Najer, M. C. Löbl, A. R. Korsch, R. Schott, S. R. Valentin, A. D. Wieck, A. Ludwig *et al.*, A bright and fast source of coherent single photons, Nat. Nanotechnol. **16**, 399 (2021).

[18] X. Ding, Y.-P. Guo, M.-C. Xu, R.-Z. Liu, G.-Y. Zou, J.-Y. Zhao, Z.-X. Ge, Q.-H. Zhang, M.-C. C. Hua-Liang Liu *et al.*, High-efficiency single-photon source above the loss-tolerant threshold for efficient linear optical quantum computing, arXiv:2311.08347.

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

[20] D. Stauffer and A. Aharony, *Introduction to Percolation Theory* (CRC Press, Boca Raton, FL, 2018).

[21] J. Wattendorff and S. Wessel, Sublattice-selective percolation on bipartite planar lattices, Phys. Rev. E **109**, 044108 (2024).

[22] M. E. J. Newman and R. M. Ziff, Efficient Monte Carlo algorithm and high-precision results for percolation, Phys. Rev. Lett. **85**, 4104 (2000).

[23] M. E. J. Newman and R. M. Ziff, Fast Monte Carlo algorithm for site or bond percolation, Phys. Rev. E **64**, 016706 (2001).

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

[25] R. J. Warburton, Single spins in self-assembled quantum dots, Nat. Mater. **12**, 483 (2013).

[26] P. Lodahl, S. Mahmoodian, and S. Stobbe, Interfacing single photons and single quantum dots with photonic nanostructures, Rev. Mod. Phys. **87**, 347 (2015).

[27] M. Atatüre, D. Englund, N. Vamivakas, S.-Y. Lee, and J. Wrachtrup, Material platforms for spin-based photonic quantum technologies, Nat. Rev. Mater. **3**, 38 (2018).

[28] P. K. Shandilya, S. Flågan, N. C. Carvalho, E. Zohari, V. K. Kavatamane, J. E. Losby, and P. E. Barclay, Diamond integrated quantum nanophotonics: Spins, photons and phonons, J. Light. Technol. **40**, 7538 (2022).

[29] A. R.-P. Montblanch, M. Barbone, I. Aharonovich, M. Atatüre, and A. C. Ferrari, Layered materials as a platform for quantum technologies, Nat. Nanotechnol. **18**, 555 (2023).

[30] S. Simmons, Scalable fault-tolerant quantum technologies with silicon color centers, PRX Quantum **5**, 010102 (2024).

[31] M. C. Löbl, Perqolate, https://github.com/nbi-hyq/perqolate (2023).

[32] S. Aaronson and D. Gottesman, Improved simulation of stabilizer circuits, Phys. Rev. A **70**, 052328 (2004).

[33] M. Rossi, M. Huber, D. Bruß, and C. Macchiavello, Quantum hypergraph states, New J. Phys. **15**, 113022 (2013).

[34] R. Stricker, D. Vodola, A. Erhard, L. Postler, M. Meth, M. Ringbauer, P. Schindler, T. Monz, M. Müller, and R. Blatt, Experimental deterministic correction of qubit loss, Nature (London) **585**, 207 (2020).

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

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

[37] K. Tiurev, P. L. Mirambell, M. B. Lauritzen, M. H. Appel, A. Tiranov, P. Lodahl, and A. S. Sørensen, Fidelity of time-bin-entangled multiphoton states from a quantum emitter, Phys. Rev. A **104**, 052604 (2021).

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

[39] N. Coste, D. Fioretto, N. Belabas, S. Wein, P. Hilaire, R. Frantzeskakis, M. Gundin, B. Goes, N. Somaschi, M. Morassi *et al.*, High-rate entanglement between a semiconductor spin and indistinguishable photons, Nat. Photon. **17**, 582 (2023).

[40] D. Cogan, Z.-E. Su, O. Kenneth, and D. Gershoni, Deterministic generation of indistinguishable photons in a cluster state, Nat. Photon. **17**, 324 (2023).

[41] H. Cao, L. M. Hansen, F. Giorgino, L. Carosini, P. Zahalka, F. Zilk, J. C. Loredo, and P. Walther, Photonic source of heralded Greenberger-Horne-Zeilinger states, Phys. Rev. Lett. **132**, 130604 (2024).

[42] N. Maring, A. Fyrillas, M. Pont, E. Ivanov, P. Stepanov, N. Margaria, W. Hease, A. Pishchagin, A. Lemaître, I. Sagnes

et al., A versatile single-photon-based quantum computing platform, Nat. Photon. **18**, 603 (2024).

[43] Y. Meng, M. L. Chan, R. B. Nielsen, M. H. Appel, Z. Liu, Y. Wang, B. Nikolai, A. D. Wieck, A. Ludwig, L. Midolo, A. Tiranov, A. S. Sørensen, and P. Lodahl, Deterministic photon source of genuine three-qubit entanglement, Nat. Commun. **15**, 7774 (2024)..

[44] A. Cabello, L. E. Danielsen, A. J. López-Tarrida, and J. R. Portillo, Optimal preparation of graph states, Phys. Rev. A **83**, 042314 (2011).

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

[46] M. Gimeno-Segovia, Towards practical linear optical quantum computing, Ph.D. thesis, Imperial College London, 2016.

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

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

[49] D. Witthaut, M. D. Lukin, and A. S. Sørensen, Photon sorters and QND detectors using single photon emitters, Europhys. Lett. **97**, 50007 (2012).

[50] H. Kesten, The critical probability of bond percolation on the square lattice equals 1/2, Commun. Math. Phys. **74**, 41 (1980).

[51] M. J. Bayerbach, S. E. D'Aurelio, P. van Loock, and S. Barz, Bell-state measurement exceeding 50% success probability with linear optics, Sci. Adv. **9**, eadf4080 (2023).

[52] S. D. Barrett and P. Kok, Efficient high-fidelity quantum computation using matter qubits and linear optics, Phys. Rev. A **71**, 060310(R) (2005).

[53] Y. L. Lim, A. Beige, and L. C. Kwek, Repeat-until-success linear optics distributed quantum computing, Phys. Rev. Lett. **95**, 030505 (2005).

[54] Y. L. Lim, S. D. Barrett, A. Beige, P. Kok, and L. C. Kwek, Repeat-until-success quantum computing using stationary and flying qubits, Phys. Rev. A **73**, 012304 (2006).

[55] L.-M. Duan and R. Raussendorf, Efficient quantum computation with probabilistic quantum gates, Phys. Rev. Lett. **95**, 080503 (2005).

[56] H. Choi, M. Pant, S. Guha, and D. Englund, Percolation-based architecture for cluster state creation using photon-mediated entanglement between atomic memories, npj Quantum Inf. **5**, 104 (2019).

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

[58] J.-O. Choi and U. Yu, Newman-Ziff algorithm for the bootstrap percolation: Application to the Archimedean lattices, J. Comput. Phys. **386**, 1 (2019).

[59] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms* (MIT Press, Cambridge, MA, 2022).

[60] R. M. Ziff, P. T. Cummings, and G. Stells, Generation of percolation cluster perimeters by a random walk, J. Phys. A: Math. Gen. **17**, 3009 (1984).

[61] K. Malarz, Random site percolation on honeycomb lattices with complex neighborhoods, Chaos: Interdiscipl. J. Nonlinear Sci. **32**, 083123 (2022).

[62] R. E. Tarjan, Efficiency of a good but not linear set union algorithm, J. ACM **22**, 215 (1975).

[63] B. A. Galler and M. J. Fisher, An improved equivalence algorithm, Commun. ACM **7**, 301 (1964).

[64] N. Delfosse and N. H. Nickerson, Almost-linear time decoding algorithm for topological codes, Quantum **5**, 595 (2021).

[65] G. de Gliniasty, P. Hilaire, P.-E. Emeriau, S. C. Wein, A. Salavrakos, and S. Mansfield, A spin-optical quantum computing architecture, Quantum **8**, 1423 (2024).

[66] M. Pompili, S. L. Hermans, S. Baier, H. K. Beukers, P. C. Humphreys, R. N. Schouten, R. F. Vermeulen, M. J. Tiggelman, L. dos Santos Martins, B. Dirkse *et al.*, Realization of a multinode quantum network of remote solid-state qubits, Science **372**, 259 (2021).

[67] M. H. Appel, A. Ghorbal, N. Shofer, L. Zaporski, S. Manna, S. F. C. da Silva, U. Haeusler, C. L. Gall, A. Rastelli, D. A. Gangloff *et al.*, Many-body quantum register for a spin qubit, arXiv:2404.19680.

[68] S. C. Van der Marck, Calculation of percolation thresholds in high dimensions for fcc, bcc and diamond lattices, Intl. J. Mod. Phys. C **09**, 529 (1998).

[69] S. E. Economou, N. Lindner, and T. Rudolph, Optically generated 2-dimensional photonic cluster state from coupled quantum dots, Phys. Rev. Lett. **105**, 093601 (2010).

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

[71] A. Acín, J. I. Cirac, and M. Lewenstein, Entanglement percolation in quantum networks, Nat. Phys. **3**, 256 (2007).

[72] T. M. Stace, S. D. Barrett, and A. C. Doherty, Thresholds for topological codes in the presence of loss, Phys. Rev. Lett. **102**, 200501 (2009).

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

[74] J. M. Auger, H. Anwar, M. Gimeno-Segovia, T. M. Stace, and D. E. Browne, Fault-tolerant quantum computation with nondeterministic entangling gates, Phys. Rev. A **97**, 030301(R) (2018).

[75] D. Vodola, D. Amaro, M. A. Martin-Delgado, and M. Müller, Twins percolation for qubit losses in topological color codes, Phys. Rev. Lett. **121**, 060501 (2018).

[76] D. Amaro, J. Bennett, D. Vodola, and M. Müller, Analytical percolation theory for topological color codes under qubit loss, Phys. Rev. A **101**, 032317 (2020).

[77] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, J. Math. Phys. **43**, 4452 (2002).

[78] M. B. Hastings, G. H. Watson, and R. G. Melko, Self-correcting quantum memories beyond the percolation threshold, Phys. Rev. Lett. **112**, 070501 (2014).

[79] M. C. Löbl, S. Paesani, A. S. Sørensen, https://doi.org/10. 17894/ucph.7a181493-1071-418b-a841-9ef5bd0d631e.

[80] M. C. Löbl, S. Paesani, A. S. Sørensen, https://doi.org/10. 17894/ucph.99e8653a-71fa-451e-92fd-4eba3dab2650.

[81] R. M. Ziff, Scaling behavior of explosive percolation on the square lattice, Phys. Rev. E **82**, 051105 (2010).

[82] N. Bastas, K. Kosmidis, P. Giazitzidis, and M. Maragakis, Method for estimating critical exponents in percolation processes with low sampling, Phys. Rev. E **90**, 062101 (2014).

[83] K. Malarz, Percolation thresholds on a triangular lattice for neighborhoods containing sites up to the fifth coordination zone, Phys. Rev. E **103**, 052107 (2021).

[84] Ł. Kurzawski and K. Malarz, Simple cubic random-site percolation thresholds for complex neighbourhoods, Rep. Math. Phys. **70**, 163 (2012).

[85] K. Malarz and S. Galam, Square-lattice site percolation at increasing ranges of neighbor bonds, Phys. Rev. E **71**, 016125 (2005).

[86] Z. Xun and R. M. Ziff, Bond percolation on simple cubic lattices with extended neighborhoods, Phys. Rev. E **102**, 012102 (2020).

[87] P. Zhao, J. Yan, Z. Xun, D. Hao, and R. M. Ziff, Site and bond percolation on four-dimensional simple hypercubic lattices with extended neighborhoods, J. Stat. Mech.: Theory Expt. (2022) 033202.

[88] M. F. Sykes and J. W. Essam, Exact critical percolation probabilities for site and bond problems in two dimensions, J. Math. Phys. **5**, 1117 (1964).

[89] X. Xu, J. Wang, J.-P. Lv, and Y. Deng, Simultaneous analysis of three-dimensional percolation models, Front. Phys. **9**, 113 (2014).

[90] J. Wang, Z. Zhou, W. Zhang, T. M. Garoni, and Y. Deng, Bond and site percolation in three dimensions, Phys. Rev. E **87**, 052107 (2013).