


Mitigating fabrication errors by recovering defective syndrome qubits in surface code

Sengthai Heng , Dongmin Kim, and Youngsun Han*

Department of AI Convergence, Pukyong National University, Busan 48513, South Korea



(Received 29 September 2023; accepted 5 December 2023; published 16 January 2024)

Fabrication errors on qubits can render data and syndrome qubits faulty, impacting quantum-processor reliability. Our proposed method focuses on the recovery of faulty syndrome qubits, preserving the functionality of neighboring qubits and minimizing the loss of data qubits. Through experimental simulation, we demonstrate that our approach significantly improves percolation rates and reduces logical error rates while enhancing the robustness of the surface code against fabrication errors. Moreover, our approach raises the critical point for syndrome-fabrication errors by 29.4% on average and 33.33% when the percolation rate reaches 100%. Furthermore, it reduces logical error rates by 18.42% on average and achieves a 50% reduction in low computational Pauli error rates.

DOI: [10.1103/PhysRevA.109.012420](https://doi.org/10.1103/PhysRevA.109.012420)

I. INTRODUCTION

Fault-tolerance quantum computers are paramount for addressing meaningful problems (e.g., Shor's algorithm [1], Grover's search algorithm [2], and simulating quantum systems [3]). These algorithms promise revolutionary advancements in cryptography [4–6], optimization [7–9], and materials science [10–12]; however, their successful implementation relies on quantum error correction (QEC) methods.

The surface code is one of the most promising QEC methods, attributed to its high threshold and relatively simple lattice structure [13–17]. In the surface code, quantum information is stored in two types of physical qubits: data qubits and syndrome qubits. Data qubits are responsible for encoding and processing the actual quantum information. Contrarily, syndrome qubits are placed around the data qubits and are crucial for error detection. These qubits interact with their neighboring data qubits and are employed to identify errors that may occur during quantum computation. However, similar to any physical implementation of quantum computation, it is susceptible to errors, particularly fabrication errors, during the manufacturing of qubits [18,19]. Fabrication errors result from the imperfect manufacturing processes inherent in the construction of quantum hardware [18,20–26]. For example, although the latest Osprey quantum processor from IBM reached 433 qubits, it has 19 (4.39%) defective qubits [27]. According to the estimates of [25], considering the current technology, it is anticipated that roughly 2% of the qubits on a transmon device exhibit faults.

Several innovative approaches for mitigating the impact of fabrication errors in the surface code have been proposed. One approach involves using sacrificial qubits to bolster the topology. By introducing additional sacrificial qubits [22], this method acts as a buffer against fabrication errors, ensuring the

integrity of the quantum processor's overall topology. Another strategy incorporates primitive SWAP gates into the construction of the syndrome readout circuit [23], offering a unique approach to error correction. In contrast, another approach [28] maintains the original construction of the surface code without attempting to modify it. Instead, it focuses on directly measuring the (defective) gauge operators [29,30], resulting in deterministic outcomes for the supercheck operators and providing an alternative means of error detection and correction. Lin *et al.* [26] proposed an automated method, working on a rotated surface code with a defective grid and generating syndrome measurement circuits. In [18], a protocol that places a shell around a group of defective data and syndrome qubits caused by cosmic rays was proposed and can be adapted to the method in [28]. However, syndrome-qubit-fabrication errors have a greater impact on the lattice compared to those affecting data qubits [28].

To minimize syndrome qubit-fabrication errors, we propose a scheme for mitigating the impact of *faulty syndrome qubits* (FSQs) in the surface code. Our method aims to recover FSQs, thereby preserving the functionality of neighboring qubits and minimizing the loss of data qubits for computation. We present our approach's architecture and implementation details, demonstrating its effectiveness through extensive simulations. The key contributions of this research can be summarized as follows:

(i) We introduce a method for recovering FSQs by reconfiguring stabilizers. The proposed approach employs bridge gates to redirect error syndromes from FSQs to nearby data qubits, thereby preserving quantum information and minimizing the loss of data qubits.

(ii) We demonstrate that the proposed method significantly enhances percolation rates, postponing the critical point for syndrome-fabrication errors by 29.4% on average and 33.33% when the percolation rate reaches 100%.

(iii) We highlight that the proposed method reduces logical error rates by 18.42% on average and achieves a remarkable 50% reduction at low computational Pauli error rates.

*youngsun@pknu.ac.kr

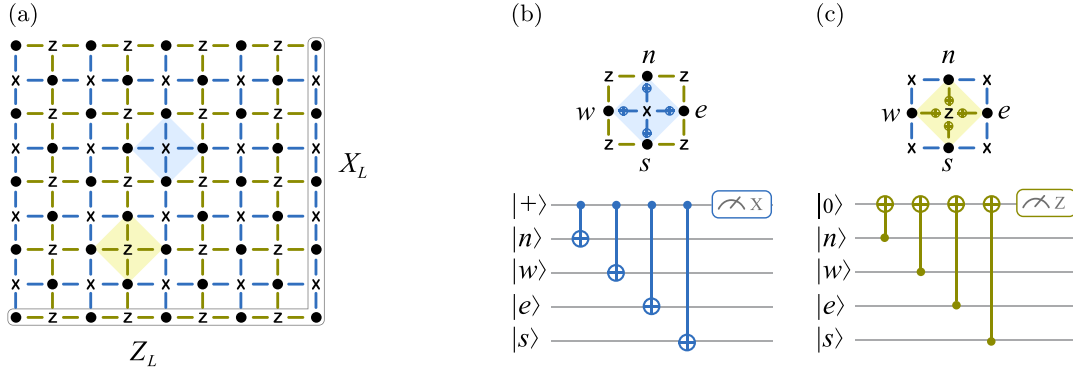


FIG. 1. Example of the lattice distance-5 surface code with the syndrome circuits for x -type and z -type stabilizer generators. The star x and plaquette z of the lattice in (a) are represented by x -type and z -type stabilizer generators in (b) and (c), respectively. The logical operators, X_L and Z_L , correspond to anticommuting string entities spanning the lattice, and each operator commutes with all the stabilizer generators.

This paper is organized as follows: Sec. II covers the surface code and fabrication errors in quantum computation. Section III describes our proposed scheme, including the architecture and recovery process for FSQs. In Sec. IV, we describe the noise model and experimental simulation. In Sec. V, we analyze percolation thresholds, effective code distances, and logical error rates, presenting the results of our numerical simulations. In Sec. VI, we conclude the paper and provide potential future directions in fault-tolerant quantum computing.

II. BACKGROUND

A. Surface code

Surface codes utilize a two-dimensional $L \times L$ lattice of entangled physical qubits to define logical qubits, as shown in Fig. 1(a), i.e., a 5×5 lattice constituting a distance-5 surface code. The distance- L code or code distance L is the measure of a code's strength, defined as the size of the shortest undetectable error chain or logical operator [31,32]. The data qubits storing the computation quantum state are depicted by black dots, and the syndrome qubits measuring the X and Z parity are represented by x and z , respectively. Each data qubit and syndrome qubit are connected by a coupler to apply a multiqubit gate, denoted by a line. The x and z syndrome qubits check the simultaneous eigenstates of their neighboring four data qubits, as shown by the blue and green backgrounds, respectively. Figures 1(b) and 1(c), corresponding to Fig. 1(a), show the standard syndrome-extraction circuits associated with the x and z syndrome qubits, respectively. These circuits are known as x - and z -type stabilizer generators. The measurement of all stabilizer generators entails a six-step process, constituting one round of syndrome measurement [33]. These steps include initializing the qubits, performing two-qubit gates four times, and measuring the syndrome qubit once.

The logical X_L and Z_L operators are anticommuting string entities traversing the lattice, demonstrating commutation with all stabilizer generators, as shown in Fig. 1(a). The effect of a logical operator on a logical state within the surface code remains unchanged when multiplied by a stabilizer generator. Consequently, numerous equivalent logical operators can be formulated. The code's distance is directly related to the minimal weight of the shortest logical operator. Thus, an ideal

surface code L is achieved [28]. Expanding the lattice size promises enhanced error protection, as a greater number of individual errors is required to induce a logical error [34].

The error-detection process involves measuring all the stabilizer generators [29]. This measurement causes any random errors in the qubits to become Pauli X and Z errors on the data qubits [35]. If there are no errors, all measurements from $|+\rangle$ and $|0\rangle$ are $+1$. If the error anticommutes with the stabilizer generator, its outcome flips to -1 . The collection of measurement outcomes obtained from the stabilizer generator measurements is commonly referred to as the syndrome [28]. Thus, those syndrome measurements are passed to a decoder, which is a classical algorithm for identifying and correcting the error that is the least probable to cause a *logical error* [36–38]. We use this logical error rate as a metric to evaluate our proposed method.

B. Fabrication error

Fabrication errors occur when physical qubits are defective during the manufacturing process or due to the emergence of faulty components during the chip's life span [39,40]. This defect is permanent, and its impact on the surface code can vary depending on the qubit type. The fabrication-error locations within the surface code are known in advance. Additionally, we assume that the qubits in surface-code chips can be turned off or disabled. There are two types of fabrication errors: qubit-fabrication errors and link-fabrication errors [25,26,28,39].

A qubit-fabrication error refers to a fault in a qubit, whether a data qubit [*faulty data qubit* (FDQ)] or a syndrome qubit (FSQ), which makes it unreliable for storing quantum information. A link-fabrication error prevents two qubits from interacting as intended, thereby hindering the execution of multiqubit gates. However, in this research, we focus on qubit-fabrication errors, particularly on a syndrome qubit.

In our terminology, as described in [28], we maintain consistency when referring to various types of failed components. Specifically, we use the term *faulty* exclusively for components that have permanent fabrication errors and the term *disabled* for components that we intentionally deactivate.

Fabrication errors pose significant damage to the construction of surface codes, as they can introduce additional degrees

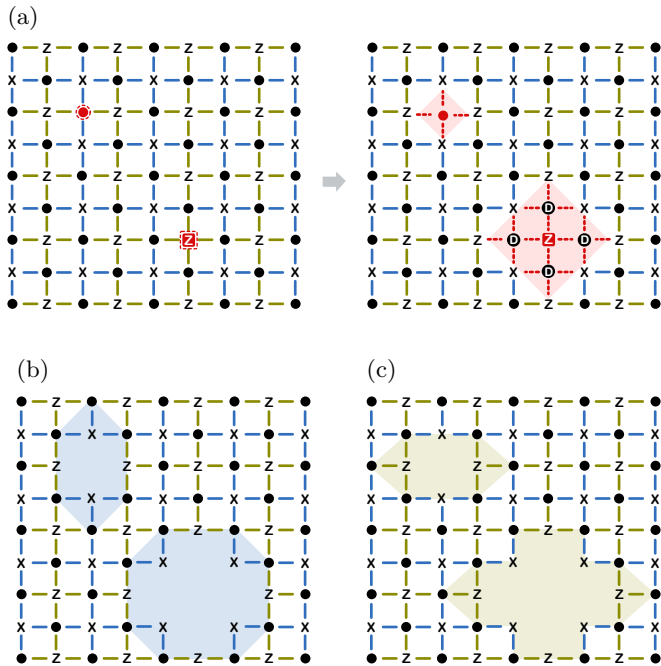


FIG. 2. (a) When a data (syndrome) qubit is inoperative or defective (indicated by a red dashed border), the associated links (comprising all data qubits involved in the corresponding stabilizer generator) are disabled. (b) Superstars are created on the x -type supercheck operators. (c) Superplaquettes are formed on the z -type supercheck operators.

of freedom, effectively creating extra logical qubits. For instance, in cases where a syndrome qubit is found to be faulty, the associated stabilizer generator might be disabled [23,28]. However, this action constructs a new logical qubit that can interact with the intended logical qubit, thereby decreasing the code distance. While reducing code distance alone may not be problematic, assuming that fabrication errors are randomly distributed across the code, the code distance will progressively decrease as the lattice size L increases. This phenomenon leads to a pseudotreshold behavior for smaller lattice sizes which becomes negligible for larger lattices [28].

Figure 2(a) shows the detrimental effect of faulty data and syndrome qubits (surrounding dashed lines) on 5×5 lattices. If a qubit is faulty, the supercheck operator [20] comes into play. This operator addresses the loss of data qubits for which the product of two stabilizer generators remains within the code stabilizer. Consequently, when a data qubit is lost, the two neighboring defective stabilizer generators can be jointly measured, forming *large supercheck operators*. Our approach, based on the methodology outlined in [28], utilizes supercheck operators and the concept of *gauge qubits* to address fabrication errors effectively. For the FDQ in Fig. 2(a), the adjacent damaged generators are anticommutative, but their supercheck operator product remains deterministic. Each *disabled data qubit* (DDQ) in the surface code raises one degree of freedom or *gauge qubit* [30,41].

The gauge qubits' logical Pauli X and Z operators, represented as the damaged x -type and z -type stabilizers (or gauge operators), are measured to randomize the gauge qubit's logical state. However, this randomization is inconsequential

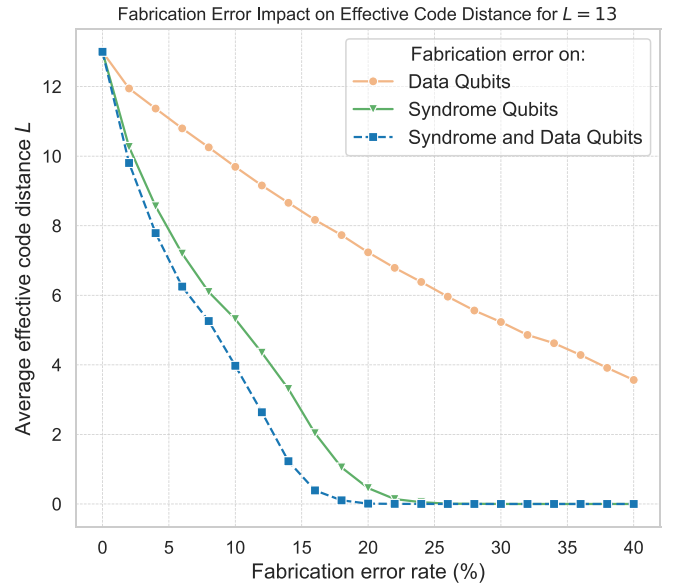


FIG. 3. Fabrication errors randomly occur on qubits, with L representing the code distance, and these errors have an impact on the effective code distance.

since the specific state of the gauge qubit does not matter. Notably, X or Z operator strings cannot terminate without detection within this region, unlike when turning off a stabilizer generator. While gauge operators are slightly lower L , they still scale proportionally to the physical lattice size.

Additionally, the product of the damaged stabilizer generators forming the supercheck operator commutes with all damaged stabilizer generators, allowing the effective use of supercheck operators for error correction during classical processing. These supercheck operators are formed by the products of damaged x -type and z -type stabilizers, as illustrated in Figs. 2(b) and 2(c), respectively.

The utilization of supercheck operators is constrained by percolation phenomena [42]. When a continuous string of FDQs percolates throughout the lattice, it becomes impossible to encode a logical qubit at the same code distance as intended. Consequently, the effective code distance decreases due to the presence of these fabrication defects [24]. However, this reduction is still manageable until the lattice reaches the percolation threshold, beyond which it becomes impractical to maintain the intended code distance.

III. PROPOSED METHOD

Figure 3 presents simulations involving the fabrication of numerous planar code lattices, each subject to different fabrication errors (see Sec. V). The effective L for each lattice is determined by determining the lowest-weight logical operator, and this process is averaged across all simulation runs. The results indicate that the fabrication errors occurring on syndrome qubits have a more severe impact than the errors limited to only data qubits. To mitigate the adverse effects of the fabrication errors on syndrome qubits, we introduce a scheme aimed at their recovery.

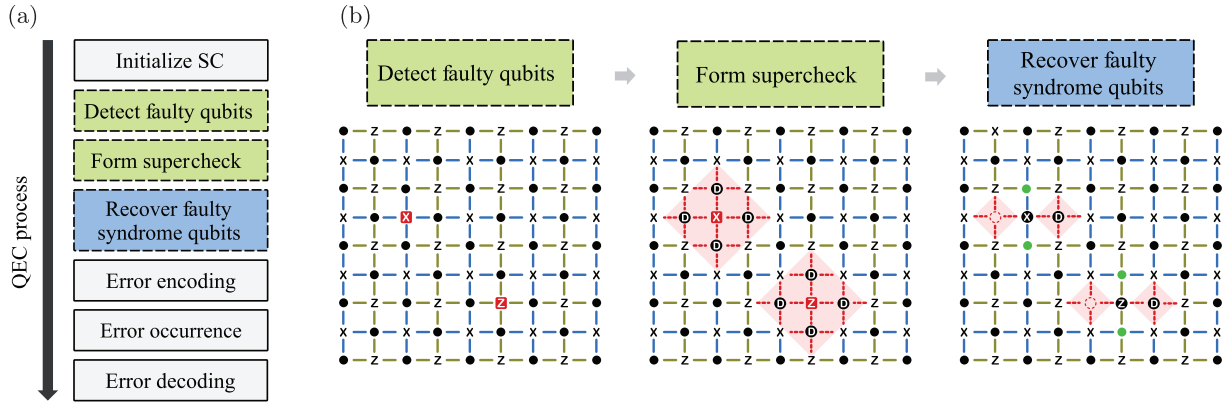


FIG. 4. (a)The QEC flow process involves the recovery of fault syndrome qubits. The surface code is initialized first, faulty qubits are detected, and superchecks are formed. The next step involves recovering faulty syndrome qubits, followed by error encoding, logical quantum computing where an error occurs, and error decoding. The processes presented in the boxes with dashed borders are designed to handle faulty qubits. (b) The QEC process detects faulty qubits, forms superchecks, and recovers faulty syndrome qubits. x - and z -type syndrome qubits in red are faulty qubits, whereas disabled qubits are labeled D . The red dots represent connections to which multiqubit gates cannot be applied.

A. Overall architecture

Figure 4(a) shows the QEC flow process described in [28], with the proposed method applied. The processes with dashed borders are methods for solving fabrication-error problems. Those enclosed within the green box adhere to the original method [28], whereas our scheme is depicted within the blue box. The QEC sequence begins with the initialization of the surface code and configured stabilizers with the intended L . Next, faulty qubits are identified even before QEC computation since the faulty fabrication qubits are known. Third, stabilizers are reconfigured, and large supercheck operators are formed from [28]. Thereafter, our scheme reconfigures the stabilizers containing FSQs from the previous method. Finally, error encoding, logical quantum computation where an error occurs, and error decoding are performed as usual.

The main idea is to reconfigure the stabilizers to ensure the continuity of QEC amid FSQs. Subsequently, those error syndromes are collected into nearby data qubits instead of the FSQs. To achieve this, we employ bridge gates [43,44], effectively redirecting the error syndromes. Essentially, this process resembles the *recovery of FSQs*.

Figure 4(b) visualizes examples of the QEC process, including the detection of faulty qubits, the construction of supercheck operators, and the recovery of FSQs. When the x - and z -type syndrome qubits depicted in red are identified as faulty, their stabilizers are reconfigured to create supercheck operations. These supercheck operations disable neighboring data qubits and links, which may reduce the L . To avoid this problem, our proposed approach leverages a DDQ as the syndrome qubit, thereby collecting error syndromes from it. This adjustment allows the activation of two adjacent data qubits highlighted in green for the storage of quantum information, as depicted in Fig. 4(b).

B. Recovery of faulty syndrome qubits

Figure 5 provides a visual representation of the circuit diagram detailing the intricate aspects of the FSQ recovery scheme. To ensure clarity and precision in our discussion, we employ two key terms [23]: *qubit device* and *qubit variable*.

A qubit device refers to the physical structure responsible for manipulating the qubit variable, including semiconductor qubits, photonic qubits, and trapped-ion qubits. Conversely, a qubit variable represents the encoded information residing on a qubit device. In Fig. 5(a), the qubit variable $(X1, X2, n, s, e, w)$ corresponds to the specific qubit device (x, z, \bullet) labeled in Fig. 5(b).

The proposed solution aims to recover the FSQs by reconfiguring the syndrome-extraction circuit to obtain the error syndrome from one of the DDQs instead. As shown in Fig. 5(b), there are four data qubits (n, s, e, w) that are chosen to be disabled, and then we can utilize one of them as a *new syndrome qubit* (NSQ). Recall from Sec. II that to extract error syndromes from data qubits, we need to apply controlled NOT (CNOT) gates from (to) data qubits to (from) syndrome qubits based on the type of stabilizer. Here, we apply the CNOT gate to the data qubit and NSQ.

Figure 5 shows the detailed reconfiguration. In this example, we utilize $|e\rangle$ as an NSQ, which needs to perform a check operation with the surrounding data qubit. However, there are constraints to consider, such as not allowing $|e\rangle$ to perform check operations on itself, as it now serves as a syndrome qubit. Additionally, the $|n\rangle$, $|s\rangle$, and $|w\rangle$ qubits are unable to directly apply CNOT gates to $|e\rangle$ due to hardware limitations.

To address these constraints, we introduce a bridge gate, allowing us to connect $|e\rangle$ to the $|X1\rangle$ and $|X2\rangle$ syndrome qubits, which are in turn connected to the $|n\rangle$ and $|s\rangle$ qubits, as shown in Fig. 5(c). We define such $|X1\rangle$ and $|X2\rangle$ syndrome qubits as *bridge syndrome qubits* (BSQs). However, qubits $|w\rangle$ and $|e\rangle$ have long-range connectivity, making the use of only a bridge gate impractical. The bridge gate's feasibility is limited to situations where the distance between qubits is confined to a single qubit length. We can apply a SWAP gate, which is similar to the method used in [23], but it increases the number of gates dramatically. To avoid this, we keep the $|w\rangle$ qubit as a disabled qubit, as in the previous configuration. Consequently, the $|n\rangle$ and $|s\rangle$ data qubits become enabled and can be used as usual, as indicated by the green dots in Fig. 5(d). We define these enabled $|n\rangle$ and $|s\rangle$ data qubits as *yield data qubits* (YDQs). Notably, this method replaces FSQs by manipulating the qubit

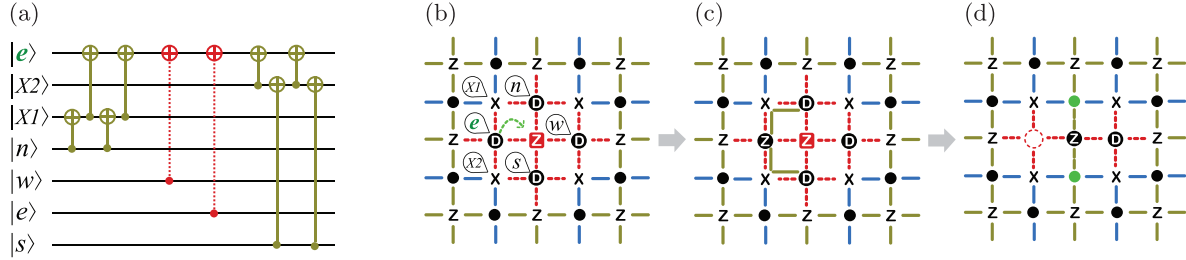


FIG. 5. The circuit diagram and surface-code lattice depict the implementation of the recovery of the faulty-syndrome-qubit scheme. The circuit in (a) that shows how to replace the faulty syndrome qubits contains qubits corresponding to the qubit labels in (b). (c) depicts the process of recovering syndrome qubits. The first step involves $|e\rangle$ qubit routing to $|n\rangle$ and $|s\rangle$ qubits using bridge gates. Next, the surface code in (d) is shown after replacement, and the result of recovering two data qubits is presented in green.

states through bridge gates rather than physically relocating them. However, to make the approach more intuitive, in Fig. 5, we show the process of $|e\rangle$ physically moving to the FSQ.

One challenge of this recovery is choosing which disabled data qubit should be the NSQ. For simplicity, one of the DDQs is selected if it is not disabled by another supercheck stabilizer and is not faulty. However, an issue arises when FSQs are located at the lattice's edge and no corresponding stabilizer generator pairs with this faulty qubit [28]. To address this situation, the lattice's edge must be redefined by entirely disabling the damaged stabilizer generator. This redefinition process may need to be repeated if any of the qubits within this new edge become faulty, effectively leading to the disabling of a supercheck operator.

For clarity, we present the pseudocode for our proposed method in Algorithm 1, which corresponds to the details provided in Fig. 5. Algorithm 1 begins by identifying FSQs, and if any of them are situated at the lattice's edge, they are left unchanged. For each eligible FSQ, we proceed to locate nearby DDQs, which, in the context of Fig. 5, correspond to $|n\rangle$, $|s\rangle$, $|e\rangle$, and $|w\rangle$.

For each of these DDQs, we evaluate whether they meet certain constraints. If any of the DDQs fulfill these constraints, one of them is selected to serve as the NSQ. If none of the DDQs meets the criteria, we proceed to another FSQ.

Once a suitable NSQ is identified or selected, the algorithm proceeds to determine the expected YDQs, in Fig. 5, corresponding to $|n\rangle$ and $|s\rangle$. For each YDQ, we find the corresponding BSQs, represented by $|X1\rangle$ and $|X2\rangle$ in Fig. 5. Finally, we enable YDQ_i and apply a bridge gate to establish connections between YDQ_i , BSQ_i , and the NSQ. This algorithm effectively captures the steps involved in the recovery of FSQs, ensuring the continuity of QEC while preserving the lattice integrity.

IV. PERFORMANCE EVALUATION

A. Error model

Computational Pauli error rates p_{comp} are applied to every gate in our model. We follow the assumption from [28] that each two-qubit gate acts perfectly without any errors in [34] and suffers from only depolarizing Pauli noise, occurring with a probability of p_{comp} . We assumed a single gate (only identity in this experiment) also acts perfectly, followed by depolarizing Pauli noise with probability $4p_{\text{comp}}/5$. The reason for this is based on [28,45]: $4p/5$ represents the error rate experienced

by each qubit participating in a two-qubit gate when exposed to depolarizing noise with a probability of p . Moreover, this assumption agrees with the experiments in [14]. It is assumed that preparation has a probability of p_{comp} to initialize the state on an orthonormal basis, whereas measurement has a probability of p_{comp} to yield an incorrect outcome.

In preparation for error correction procedures, all fabrication errors are initially treated as distinct and unrelated occurrences. We assume that the precise locations of these fabrication errors have already been pinpointed. In this sce-

Algorithm 1. Reconfiguration of the stabilizer to recover defective syndrome qubits.

```

// Faulty Syndrome Qubits
1  $FSQs \leftarrow \text{GetFaultySyndQubits}();$ 
2 for  $FSQ$  in  $FSQs$  do
3   if  $FSQ$  is at lattice edge then
4     continue;
5   end
6   // Disable Data Qubits
7    $DDQs \leftarrow \text{GetNearbyDisabledDataQubits}(FSQ);$ 
8   // New Syndrome Qubit
9    $NSQ \leftarrow \text{None};$ 
10  for  $DDQ$  in  $DDQs$  do
11    if  $DDQ$  is faulty then
12      continue;
13    end
14    // If DDQ is disabled by another
15    // stabilizer
16    if  $DDQ.\text{number\_disabled}() \geq 2$  then
17       $NSQ \leftarrow DDQ;$ 
18      break;
19    end
20  end
21  if  $NSQ$  is None then
22    continue;
23  end
24  // Yield Data Qubits
25   $YDQs \leftarrow \text{GetYieldDataQubits}(DDQ, NSQ);$ 
26  for  $YDQ$  in  $YDQs$  do
27    // Bridge Syndrome Qubit
28     $BSQ \leftarrow \text{FindBridgeSyndQubit}(YDQ, NSQ);$ 
29     $YDQ.\text{set\_enable}();$ 
30     $\text{ApplyBridgeGate}(YDQ, BSQ, NSQ);$ 
31  end
32 end

```

nario, every qubit, whether it is a syndrome qubit or a data qubit, faces a fabrication error, and this likelihood is denoted by the parameter p_{comp} . Similarly, every connection or link is prone to fabrication errors with a probability indicated by p_{link} . p_{qubit} indicates fabrication-error rate. These parameters, namely, p_{comp} , p_{qubit} , and p_{link} , can be adjusted independently. However, for this research, we exclusively focus on analyzing the parameter p_{qubit} , making the probability p_{link} almost negligible. Consequently, we assume that the errors in couplers, which are essential for carrying out two-qubit gates, are inherently presumed to be absent.

B. Experimental simulation

Our numerical simulation is implemented based on the simulator from [28]. First, the simulation generates a surface-code lattice consisting of qubits with fabrication-error rates p_{qubit} . The fabrication errors are on the data and syndrome qubits, followed by the qubits or links for forming the supercheck operators. The logical operators are constructed through a path-finding algorithm operating on the x -type and z -type lattices. In cases where a logical operator cannot be identified, the lattice undergoes a percolation process involving FDQs, or DDQs, leading to the termination of the simulation.

When a logical operator is constructed, the simulation executes $2 \times L$ cycles of syndrome measurement. Each cycle comprises the initialization of syndrome qubits, four steps of two-qubit gates, and subsequent syndrome-qubit measurement. These processes are each allotted a single unit of time. Any qubit not engaged in a two-qubit gate, measurement, or initialization during a specific time step experiences an identity gate operation.

To initiate the code, an initial cycle of flawless x -type and z -type measurements is conducted to obtain error-free results for each measurement. The simulation concludes with a final cycle of perfect measurements. For all other stabilizer-measurement cycles, the provided Pauli error model is applied.

We adopt a stabilizer approach identical to that presented in [28], employing the CNOT-Hadamard-phase stabilizer algorithm [46]. This methodology is utilized to simulate the quantum state during gate operations and measurements, thereby verifying the accuracy of the gauge operators' results. Following the acquisition of all measurements, a minimum-weight perfect-matching routine, involving the Blossom V algorithm [47], is employed to ascertain the necessary correction based on the acquired syndrome information.

The edge weights utilized for the perfect-matching routine are determined using the methodologies outlined in [28,48], optimizing the matching process. Syndrome measurement is emulated on x -type and z -type lattices, although error correction is executed solely on one lattice to minimize computational overhead. This decision is informed by the symmetry between the x -type and z -type lattices, resulting in nearly identical logical error rates. After the correction, an assessment is conducted to detect the presence of a logical X error by verifying if the combined error and correction sequences commute with the logical Z operator.

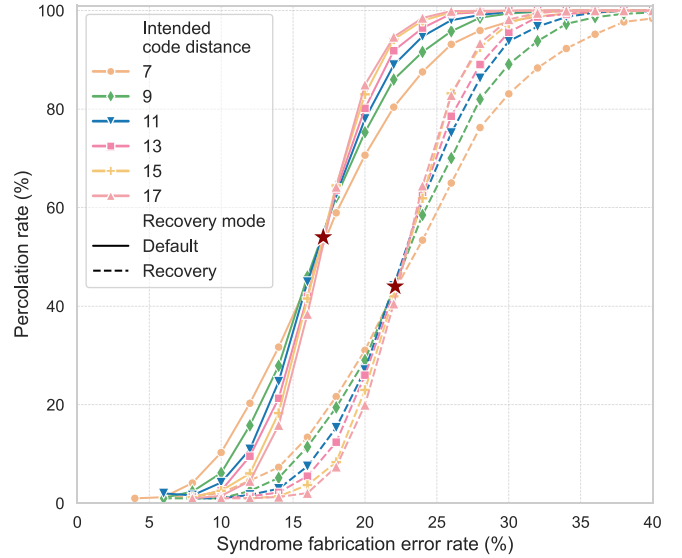


FIG. 6. Percolation rates for the syndrome-fabrication-error rate. The percolation rates, with default representing the method from prior research and recovery denoting our proposed approach in which the recovery method is applied. A star is used to denote the critical points for the percolation error rate. Specifically, the critical point for the default method occurs at a syndrome-fabrication-error rate of 17%, whereas the recovery method is observed at 22%.

The Pauli error rates are systematically adjusted, spanning $p_{\text{comp}} = 0.05\%$ to $p_{\text{comp}} = 1.00\%$ in increments of 0.05% . Each fabrication-error rate is modulated from 0% to 40% at 2% intervals. Moreover, the fabrication error will randomly occur on the surface code. Using the configuration above, the experiment is conducted following [28]. Each configuration runs 100 times to obtain an average result. Additionally, 15×10^5 iterations are simulated for each combination of error rates and code distances from distance-7 to distance-17 codes.

V. EXPERIMENTAL RESULTS

A. Percolation thresholds

Our primary result is illustrated in Fig. 6, which illustrates the percolation behavior of the surface code concerning the syndrome-fabrication-error rate, under the assumption of the absence of data-fabrication errors. The percolation rate varies from 0% to 100% , signifying the rate at which the effective code distance reaches zero. In Fig. 6, we present the results obtained using the method described in [28], labeled *Default*, and our method, referred to as *Recovery*.

The recovery method achieves a 33.33% (on average) improvement in resilience against syndrome-fabrication errors compared with the results of the default method when the percolation rate is 100% . Notably, a percolation rate of 50% serves as the bond percolation threshold analyzed in [28], and our method enhances the resilience of syndrome-fabrication errors by 27% on average in this scenario.

Moreover, Fig. 6 shows the critical point (stars) of the percolation rate. Our findings reveal that a large L corresponds to high resilience compared with a small L when operating

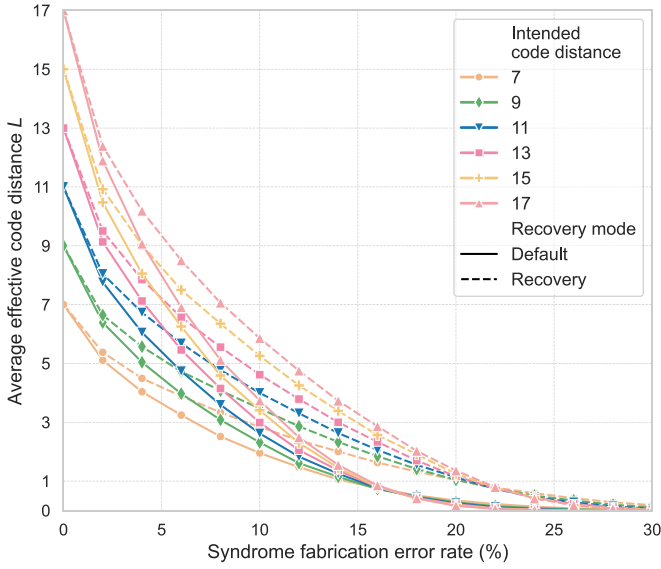


FIG. 7. Average effective code distance for the syndrome-qubit-fabrication errors using the default and proposed recovery methods.

below the critical point. Conversely, when operating above the critical point, a smaller L demonstrates superior resilience. This observation underscores the intricate relationship between L and the percolation rate, particularly in the context of the critical point within percolation theory. For the default method, the critical point occurs at a syndrome-fabrication-error rate of 17%, whereas for the recovery method, it shifts to 22%. Thus, our scheme enhances the critical point along the x axis by 22.7%, indicating that it requires a 29.4% higher syndrome-fabrication-error rate to reach the critical point.

B. Effective code distance

Figure 7 shows an analysis of the variation in the average effective code distance with the syndrome-qubit-fabrication errors for the default and proposed recovery methods. To determine the effective code distance for each lattice, we identify the lowest-weight logical operator:

$$\bar{L}' = \frac{\sum_{i=1}^n \min(L'z_i, L'x_i)}{n}. \quad (1)$$

Specifically, \bar{L}' is the average effective code distance, and n is the total number of runs. $L'z$ and $L'x$ are defined as the minimum weights of a nontrivial logical operator for z -type and x -type stabilizers. This is equivalent to the product of a logical operator on the encoded qubit and a gauge qubit operator.

Subsequently, the effective code distance is calculated by averaging the results across all simulation runs. The average effective code distance drops from the intended code distance to 1 when the syndrome-fabrication-error is 14.8% on average for the default method. Conversely, for the recovery method, it decreases the code distance to 1 when the syndrome-fabrication-error rate averages 21.8%. This indicates that our method significantly reduces the decline in the effective code distance by an average of 47.29%.

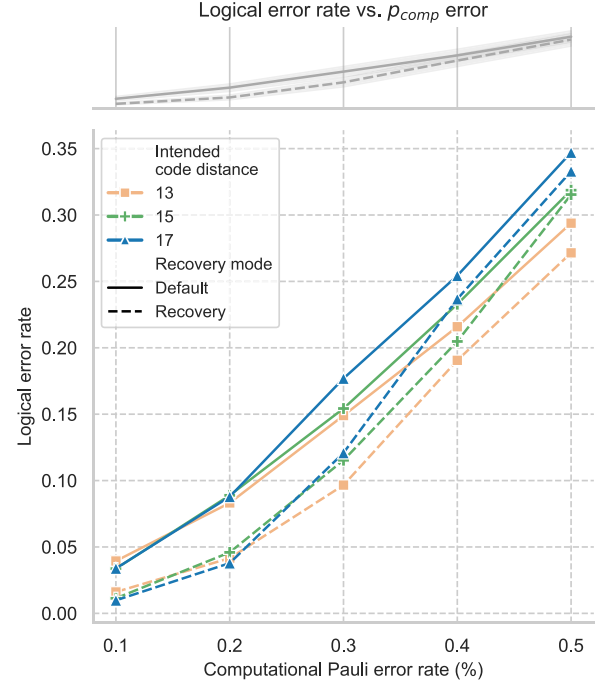


FIG. 8. Average logical error rates with the qubit-fabrication-error rate varying from 0% to 10% for intended distance-13, -15, and -17 codes. The graph above shows the average result of the intended code distance below.

C. Logical error rate

The logical error rate is calculated by dividing the number of logical errors by the total number of runs, excluding those affected by percolation. Figure 8 shows the logical error rate that is impacted by the computational Pauli error rate p_{comp} , which applies qubit-fabrication errors p_{qubit} ranging 0% to 10%. The dashed lines indicate the experiment conducted using our proposed recovery method, whereas the solid lines are those conducted using the default method from [28]. Compared to the default method, the recovery method reduced the logical qubit error rate to around 50% when $p_{\text{comp}} = 0.2\%$ and 18.42% on average. This reduction occurs because when the lattice gains more data qubits, it becomes more resilient to the error. At the crossing point, a threshold of approximately 0.2% of p_{comp} is observed for each intended code distance. This indicates that the penalty of the surface code for having a qubit-fabrication error above the cross point increases the logical error rate at high L values.

VI. CONCLUSION

In this study, we have introduced an approach for mitigating the impact of FSQs on the surface code. Our proposed method focuses on recovering FSQs, thereby preventing the disabling of neighboring qubits and minimizing the loss of data qubits available for computation.

A distinctive aspect of our approach involves the use of bridge gates for reconfiguring stabilizers and redirecting error syndromes from FSQs to nearby data qubits, which can be performed without requiring any supplementary hardware components. Moreover, using bridge gates proves to be more efficient than methods utilizing SWAP gates in [23].

Through the experimental simulation, we have demonstrated our approach's efficacy in improving the surface code's resilience against syndrome-fabrication errors. Our results indicate that the recovery method substantially enhances the percolation rate compared with the default method, effectively raising the critical point for syndrome-fabrication errors by 29.4% on average and 33.33% when the percolation rate reaches 100%. This implies that our approach allows for a more robust surface-code implementation, particularly when fabrication errors are prevalent. Furthermore, our method remarkably achieves a 50% reduction in the logical error rate at low computational Pauli error rates ($p_{\text{comp}} = 0.2\%$) and achieves 18.42% on average, compared with the results of the default method. Further, we agree with [28] that increasing the size of a surface code is beneficial only if it does not raise the fabrication-error rate, and we found that the main cause is particularly the syndrome-fabrication-error rate.

This study prompts further investigation into optimal strategies for recovering defective syndrome qubits across

various quantum error correction codes. It aims to tailor these strategies to maximize recovery efficiency while minimizing additional resource consumption. These inquiries seek to explore the theoretical limits and trade-offs associated with recovery operations, facilitating approaches to optimize the proposed method's performance in mitigating fabrication errors on qubits.

ACKNOWLEDGMENTS

This research was partly supported by Quantum Computing based Quantum Advantage challenge research (RS-2023-00257994) through the National Research Foundation of Korea (NRF) funded by the Korean government (MSIT) and an Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT; Grant No. 2020-0-00014, "A Technology Development of Quantum OS for Fault-tolerant Logical Qubit Computing Environment").

-
- [1] P. Shor, Algorithms for quantum computation: Discrete logarithms and factoring, in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science, NW Washington, DC* (IEEE Computer Society, Piscataway, NJ, 1994), pp. 124–134.
- [2] L. K. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96* (ACM Press, Philadelphia, 1996), pp. 212–219.
- [3] I. M. Georgescu, S. Ashhab, and F. Nori, Quantum simulation, *Rev. Mod. Phys.* **86**, 153 (2014).
- [4] S. S. Panda, P. A. A. Yasir, and C. M. Chandrashekar, Quantum direct communication protocol using recurrence in k -cycle quantum walks, *Phys. Rev. A* **107**, 022611 (2023).
- [5] E. Gouzien, D. Ruiz, F.-M. Le Régent, J. Guillaud, and N. Sangouard, Performance analysis of a repetition cat code architecture: Computing 256-bit elliptic curve logarithm in 9 hours with 126 133 cat qubits, *Phys. Rev. Lett.* **131**, 040602 (2023).
- [6] C. Portmann and R. Renner, Security in quantum cryptography, *Rev. Mod. Phys.* **94**, 025008 (2022).
- [7] N. N. Hegade, X. Chen, and E. Solano, Digitized counterdiabatic quantum optimization, *Phys. Rev. Res.* **4**, L042030 (2022).
- [8] J. R. McClean, M. P. Harrigan, M. Mohseni, N. C. Rubin, Z. Jiang, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Low-depth mechanisms for quantum optimization, *PRX Quantum* **2**, 030312 (2021).
- [9] M.-T. Nguyen, J.-G. Liu, J. Wurtz, M. D. Lukin, S.-T. Wang, and H. Pichler, Quantum optimization with arbitrary connectivity using Rydberg atom arrays, *PRX Quantum* **4**, 010316 (2023).
- [10] N. Yoshioka, T. Sato, Y. O. Nakagawa, Y.-Y. Ohnishi, and W. Mizukami, Variational quantum simulation for periodic materials, *Phys. Rev. Res.* **4**, 013052 (2022).
- [11] K. Nawa, T. Suzuki, K. Masuda, S. Tanaka, and Y. Miura, Quantum annealing optimization method for the design of barrier materials in magnetic tunnel junctions, *Phys. Rev. Appl.* **20**, 024044 (2023).
- [12] F. Machado, E. A. Demler, N. Y. Yao, and S. Chatterjee, Quantum noise spectroscopy of dynamical critical phenomena, *Phys. Rev. Lett.* **131**, 070801 (2023).
- [13] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Towards practical large-scale quantum computation, *Phys. Rev. A* **86**, 032324 (2012).
- [14] Y. Zhao *et al.*, Realization of an error-correcting surface code with superconducting qubits, *Phys. Rev. Lett.* **129**, 030501 (2022).
- [15] A. N. Cleland, An introduction to the surface code, *SciPost Phys. Lect. Notes* **49**, 49 (2022).
- [16] R. Versluis, S. Poletto, N. Khammassi, B. Tarasinski, N. Haider, D. J. Michalak, A. Bruno, K. Bertels, and L. DiCarlo, Scalable quantum circuit and control for a superconducting surface code, *Phys. Rev. Appl.* **8**, 034021 (2017).
- [17] A. deMarti iOlius, J. E. Martinez, P. Fuentes, P. M. Crespo, and J. Garcia-Frias, Performance of surface codes in realistic quantum hardware, *Phys. Rev. A* **106**, 062428 (2022).
- [18] A. Strikis, S. C. Benjamin, and B. J. Brown, Quantum computing is scalable on a planar array of qubits with fabrication defects, *Phys. Rev. Appl.* **19**, 064081 (2023).
- [19] A. Bilmes, A. Megrant, P. Klimov, G. Weiss, J. M. Martinis, A. V. Ustinov, and J. Lisenfeld, Resolving the positions of defects in superconducting quantum bits, *Sci. Rep.* **10**, 3090 (2020).
- [20] 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).
- [21] A. Paler, A. G. Fowler, and R. Wille, Reliable quantum circuits have defects, *XRDS: Crossroads, ACM Mag. Stud.* **23**, 34 (2016).
- [22] Y.-C. Tang and G.-X. Miao, Robust surface code topology against sparse fabrication defects in a superconducting-qubit array, *Phys. Rev. A* **93**, 032322 (2016).
- [23] S. Nagayama, A. G. Fowler, D. Horsman, S. J. Devitt, and R. V. Meter, Surface code error correction on a defective lattice, *New J. Phys.* **19**, 023050 (2017).

- [24] A. Siegel, A. Strikis, T. Flatters, and S. Benjamin, Adaptive surface code for quantum error correction in the presence of temporary or permanent defects, *Quantum* **7**, 1065 (2023).
- [25] K. N. Smith, G. S. Ravi, J. M. Baker, and F. T. Chong, Scaling superconducting quantum computers with chiplet architectures, in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Chicago, Illinois (IEEE Press, Piscataway, NJ, 2022), pp. 1092–1109.
- [26] S. F. Lin, J. Vizslai, K. N. Smith, G. S. Ravi, C. Yuan, F. T. Chong, and B. J. Brown, Empirical overhead of the adapted surface code on defective qubit arrays, [arXiv:2305.00138](https://arxiv.org/abs/2305.00138).
- [27] IBM Quantum, <https://quantum-computing.ibm.com/services/resources>.
- [28] J. M. Auger, H. Anwar, M. Gimeno-Segovia, T. M. Stace, and D. E. Browne, Fault-tolerance thresholds for the surface code with fabrication errors, *Phys. Rev. A* **96**, 042316 (2017).
- [29] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, *J. Math. Phys.* **43**, 4452 (2002).
- [30] O. Higgott and N. P. Breuckmann, Subsystem codes with high thresholds by gauge fixing and reduced qubit overhead, *Phys. Rev. X* **11**, 031039 (2021).
- [31] D. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, Surface code quantum computing by lattice surgery, *New J. Phys.* **14**, 123011 (2012).
- [32] A. Paetznick, C. Knapp, N. Delfosse, B. Bauer, J. Haah, M. B. Hastings, and M. P. da Silva, Performance of planar Floquet codes with Majorana-based qubits, *PRX Quantum* **4**, 010310 (2023).
- [33] C. Ryan-Anderson, J. G. Bohnet, K. Lee, D. Gresh, A. Hankin, J. P. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. C. Brown, T. M. Gatterman, S. K. Halit, K. Gilmore, J. A. Gerber, B. Neyenhuis, D. Hayes, and R. P. Stutz, Realization of real-time fault-tolerant quantum error correction, *Phys. Rev. X* **11**, 041058 (2021).
- [34] Google Quantum AI *et al.*, Suppressing quantum errors by scaling a surface code logical qubit, *Nature (London)* **614**, 676 (2023).
- [35] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. (Cambridge University Press, Cambridge, 2010).
- [36] Y.-H. Liu and D. Poulin, Neural belief-propagation decoders for quantum error-correcting codes, *Phys. Rev. Lett.* **122**, 200501 (2019).
- [37] S. C. Smith, B. J. Brown, and S. D. Bartlett, Local predecoder to reduce the bandwidth and latency of quantum error correction, *Phys. Rev. Appl.* **19**, 034050 (2023).
- [38] J. Fujisaki, H. Oshima, S. Sato, and K. Fujii, Practical and scalable decoder for topological quantum error correction with an Ising machine, *Phys. Rev. Res.* **4**, 043086 (2022).
- [39] J. B. Hertzberg, E. J. Zhang, S. Rosenblatt, E. Magesan, J. A. Smolin, J.-B. Yau, V. P. Adiga, M. Sandberg, M. Brink, J. M. Chow, and J. S. Orcutt, Laser-annealing Josephson junctions for yielding scaled-up superconducting quantum processors, *npj Quantum Inf.* **7**, 129 (2021).
- [40] J. M. Kreikebaum, K. P. O’Brien, A. Morvan, and I. Siddiqi, Improving wafer-scale Josephson junction resistance variation in superconducting quantum coherent circuits, *Supercond. Sci. Technol.* **33**, 06LT02 (2020).
- [41] A. Paetznick and B. W. Reichardt, Universal fault-tolerant quantum computation with only transversal gates and error correction, *Phys. Rev. Lett.* **111**, 090505 (2013).
- [42] S. D. Barrett and T. M. Stace, Fault tolerant quantum computation with very high threshold for loss errors, *Phys. Rev. Lett.* **105**, 200502 (2010).
- [43] M. Y. Siraichi, V. F. D. Santos, C. Collange, and F. M. Q. Pereira, Qubit allocation as a combination of subgraph isomorphism and token swapping, *Proc. ACM Program. Languages* **3**, 1 (2019).
- [44] T. Itoko, R. Raymond, T. Imamichi, and A. Matsuo, Optimization of quantum circuit mapping using gate transformation and commutation, *Integration* **70**, 43 (2020).
- [45] E. Knill, Quantum computing with realistically noisy devices, *Nature (London)* **434**, 39 (2005).
- [46] S. Aaronson and D. Gottesman, Improved simulation of stabilizer circuits, *Phys. Rev. A* **70**, 052328 (2004).
- [47] V. Kolmogorov, Blossom V: A new implementation of a minimum cost perfect matching algorithm, *Math. Prog. Comp.* **1**, 43 (2009).
- [48] S. Bravyi and A. Vargo, Simulation of rare events in quantum error correction, *Phys. Rev. A* **88**, 062308 (2013).