Multiplexed Quantum Communication with Surface and Hypergraph Product Codes

Shin Nishio[1,2,3], Nicholas Connolly[2], Nicolò Lo Piparo[2], William John Munro[2,3], Thomas Rowan Scruby[2], and Kae Nemoto[2,3]

[1]SOKENDAI (The Graduate University for Advanced Studies), 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan
[2]Okinawa Institute of Science and Technology Graduate University, Onna-son, Kunigami-gun, Okinawa, 904-0495, Japan
[3]National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan

arXiv:2406.08832v2 [quant-ph] 16 Jan 2025

Connecting multiple processors via quantum interconnect technologies could help overcome scalability issues in single-processor quantum computers. Transmission via these interconnects can be performed more efficiently using quantum multiplexing, where information is encoded in high-dimensional photonic degrees of freedom. We explore the effects of multiplexing on logical error rates in surface codes and hypergraph product codes. We show that, although multiplexing makes loss errors more damaging, assigning qubits to photons in an intelligent manner can minimize these effects, and the ability to encode higher-distance codes in a smaller number of photons can result in overall lower logical error rates. This multiplexing technique can also be adapted to quantum communication and multimode quantum memory with high-dimensional qudit systems.

1 Introduction

Quantum computers are expected to solve problems that are intractable using classical computation [1, 2], but these powerful quantum algorithms require high qubit counts and deep circuits to solve problems of interesting size [3, 4]. While the qubit counts of quantum processors have been increasing rapidly in recent years, various physical constraints impose limits on the possible size of a single quantum processor [5, 6]. *Quantum interconnects* provide a resolution to this problem by allowing for the networking and cooperative operation of multiple quantum processors [7], as well as the use of separate quantum memories [8], quantum repeaters [9–12] and networks [13–15] in analogy with classical computing architectures. Optical systems are considered leading candidates for practical implementation of quantum interconnects [16] and also as quantum memories [17] due to their long coherence times [18].

Due to the high noise levels inherent in quantum systems, large-scale quantum algorithms cannot be executed reliably without the use of quantum error correcting codes (QECCs) [19,20], which enable fault-tolerant quantum computation (FTQC) [21,22]. Similarly, optical interconnects [7,16] can suffer from high photon loss rates and so QECCs should be used to protect information transmitted through these channels. In principle, it may be preferable to use different codes for these different settings [23], but in practice, the transfer of information between these different codes may be challenging enough that it is easier to use only a single code. For instance, fault-tolerant logic with surface codes [24] has been very well studied [25, 26], while quantum Reed-Solomon codes [27] provide efficient and loss-tolerant protection for transmission through optical channels, but it is not clear how to interface or switch between these two families of codes. Therefore, for performing distributed computation with interconnects, using surface codes (or their generalizations) for both computation and transmission [28] is a natural alternative to using multiple codes. This is much less efficient in the sense of error-correction capability for the

Shin Nishio: parton@nii.ac.jp
Nicholas Connolly: nicholas.connolly@oist.jp
Thomas Rowan Scruby: thomas.scruby@oist.jp
Kae Nemoto: kae.nemoto@oist.jp

communication part, but these overheads can be reduced using *quantum multiplexing* [29].

Quantum multiplexing is a technique for encoding high-dimensional quantum information into a single photon by exploiting multiple different photonic degrees of freedom (DOF) or a single multi-component degree of freedom. Such encodings can be performed using only linear optical elements and can significantly reduce the resources associated with quantum communication [30–32]. In this work, we examine the potential of quantum multiplexing for enabling efficient transmission of surface and hypergraph product (HGP) codes through optical channels. Densely encoding many of these codes' qubits into small numbers of photons has the potential to make loss errors much more damaging, but we present various techniques (e.g. optimized strategies for qubit-to-photon assignment) that can mostly or completely eliminate these downsides.

The rest of this paper is organized as follows. In Sec. 2 we review the relevant background for quantum multiplexing and communication over lossy optical channels. Then in Sec. 3 we propose three approaches to error-corrected quantum communication using multiplexing that provide different ways of reducing the impact of loss errors. This is followed up in Sec. 4 and Sec. 5 with an examination of some of these approaches in more detail for surface codes and HGP codes, respectively. Finally, we discuss our findings and conclude this work in Sec. 6.

2 Background

In this section, we provide a brief overview of quantum multiplexing and erasure correction and show how these elements appear in practical quantum communication protocols.

2.1 Quantum Multiplexing

This subsection outlines the concept of quantum multiplexing and illustrates its possible implementation with an example.

In photon-based quantum information processing, various DOFs can be utilized to encode qubits. Polarizations [33], time-bins [34–36], paths (dual rail) [37], orbital angular momentum [38], and frequency-bin [39, 40] are typical examples of DOFs in a single photon which are commonly used in experiments. Multi-level time-bins make it especially easy to encode high-dimensional quantum information in a single photon. For instance, Fig. 1 (A) shows a method for encoding higher dimensional information (dimension 2^2 for two qubits) using polarization and time-bin DOF. The series of Half-Quarter-Half waveplates as shown in Fig. 1 (A) as HQH allows one to apply an arbitrary unitary gate to the polarization DOF of a photon. This corresponds to applying rotation gates around a Z-X-Z basis in a qubit. We label the horizontal and vertical components of a polarized photon with H and V, respectively, and the short and long components of a temporal mode (time-bin) with S and L, respectively.

This circuit takes a photon whose polarization is encoded with quantum information as input. This input photon has one qubit of information. After passing through this circuit, the photonic state $a\,|HS\rangle + b\,|HL\rangle + c\,|VS\rangle + d\,|VL\rangle$ has both polarization and time-bin degrees of freedom.

The polarization encodes a 2-dimensional Hilbert space, and the time-bin encodes another 2-dimensional Hilbert space. Therefore, the Hilbert space of the final encoded state has dimension 4, thus encoding 2 qubits of information. This encoding can easily be generalized to higher dimensional multiplexed photons as shown in Appendix B. Significantly, encoding high-level time-bin states only requires linear optical elements and classical optical switches.

Quantum multiplexing [29] is a method to encode higher dimensional quantum information into a single photon using these multiple degrees of freedom. One can transfer the state of this multiplexed photon embedded with an extra DOF (time-bin) into two matter qubits [29] as shown in Fig. 1 (B). First, the photon is entangled with the internal DOFs of a matter qubit, depicted as a red dot in a box in the Figure. Then the HL and VS components of the photon are switched by the switching setup module (Sw). This is a slightly modified setup in [29] where we use conventional PBSs rather than 45°-PBSs. Next, the photon interacts with another matter qubit. Here we label the basis states of a matter qubit $|0\rangle$ and $|1\rangle$. After measuring the DOFs of the photon, the state of the photon is projected into the matter qubits upon a Pauli frame correction depending on the measurement outcome (write-in). This proce-

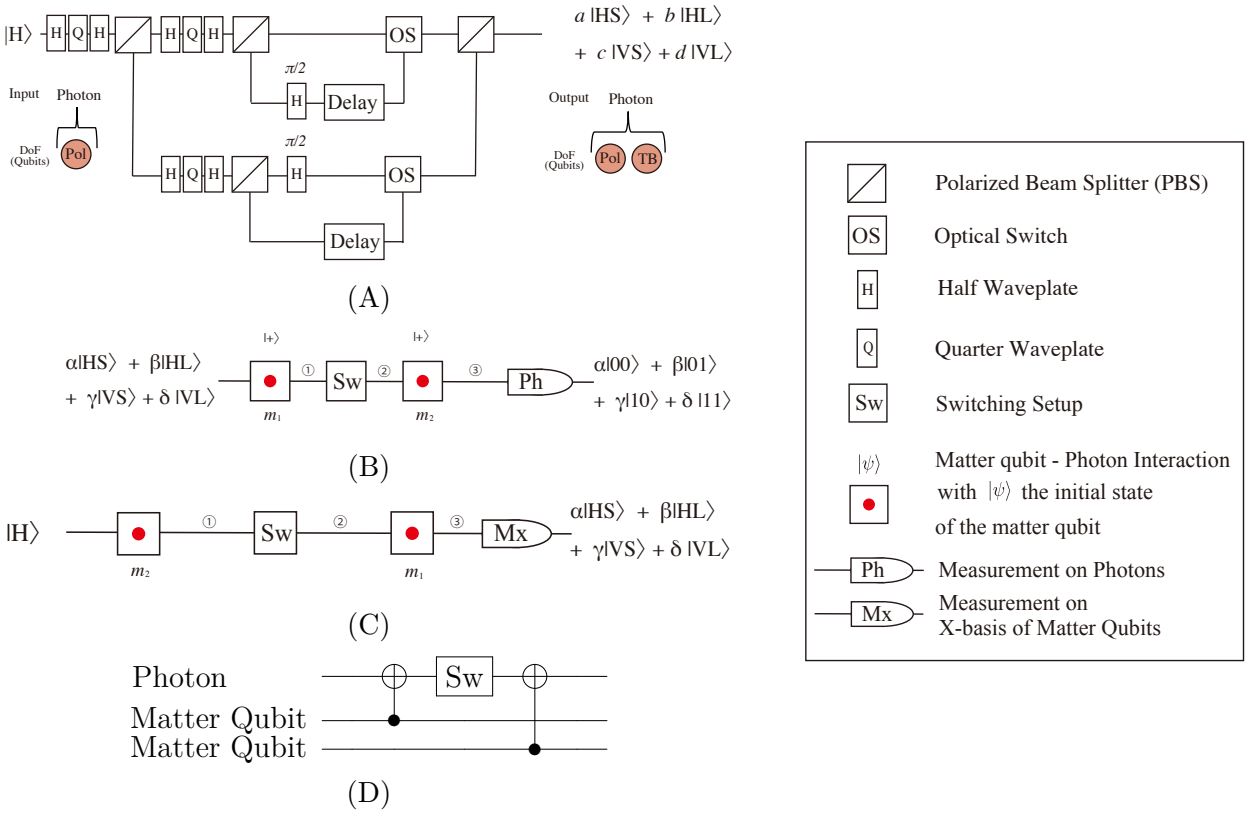


Figure 1: An example of an optical circuit encoding 2^2-dimensional quantum information into a single photon. A horizontal (H) polarized photon is sent through a half-quarter-half wave plate appropriately rotated to generate a general polarized qubit. Then a polarizing beam splitter (PBS) divides the components of such a qubit into two spatial modes. Specifically, the horizontal component is transmitted whereas the vertical component is reflected. A delay is added to these components, and two optical switches and another PBS will recombine all the components into a single spatial mode. (B) A heralding method to transfer a general state of a multiplexed photon into two matter qubits m_1 and m_2 (red dots into boxes) by using a switching setup (Sw) module, which has been used in [29]. (C) A heralding method to transfer a general state of two atoms into a multiplexed photon. Here a polarized photon interacts with an atom m_1 and then goes through a Sw module to add time-bin modes. Then this photon interacts with another atom m_2. We use $\alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle$ for the input state of two-qubit memory. Upon a successful measurement in the X-basis of the two atoms, their state will be transferred into the multiplexed photon. (D) The equivalent circuit of the setups in (B) ((C)), respectively. The interaction of a polarized photon with matter qubit m_1 (m_2) and then with matter qubit m_2 (m_1) corresponds to a CNOT gate in which the atom is the control qubit and the polarization DOF of the photon is the target qubit. Although not explicitly shown, at the end of the circuit the photon and matter qubits are measured for the setup in (B) and (C), respectively.

dure can be extended to photons with an arbitrary number of time-bin components by adding the corresponding number of matter qubits. We can recover the information stored in the matter qubits with the process shown in Fig. 1 (C) (readout). This process can be regarded as the inverse of Fig. 1 (B). Here, a polarized photon interacts with a matter qubit m_2. Next, the photon goes through the Sw module which switches the HL component and the VS component. Then, the photon interacts with another matter qubit m_1. Finally, both matter qubits are measured in the X-basis. This projects the state of the matter qubits into the multiplexed photon. We give a quantum circuit representation in Fig. 1 (D), and the full description of the state transition in the Appendix A for Fig. 1 (B) and (C).

In this work, we consider encoding 2^m-dimensional quantum information using m components of a DOF per photon where m is an integer ($m = 1$ corresponds to no multiplexing).

It is worth noticing that while quantum multiplexing allows for efficient communication, it also changes the error model. In fact, in a lossy communication channel, the loss of a photon causes the simultaneous loss of multiple qubits encoded in that photon. This can be very detrimental to the performance of this system. However, in the next section, we will devise several strategies for qubit assignment to mitigate the effects of the loss of qubits.

2.2 Erasure Channel and Correction

Let us now describe the erasure channel and decoding, which will play an important role in the quantum communication protocol.

In photonic systems, the erasure error is a localized loss error of a photon due to imperfections in the photon source, the physical channel used for its transmission, and detectors. This is the dominant source of errors in optical systems [41, 42]. Moreover, theoretical [43–46] and experimental [47–51] works have been proposed on methods to map errors from different sources to erasure errors in multiple physical systems recently. Therefore, correction of erasure errors is of engineering importance because it can be applied in a variety of systems where erasure errors are not the main source of error.

The erasure channel is given by

$$\rho \to (1-\varepsilon)\rho + \varepsilon \left|e\right\rangle \left\langle e\right| \tag{1}$$

where $|e\rangle$ indicates the erased state, and ε is the probability of erasure. Due to the fact that the erased state is not in the original Hilbert space, it is possible to detect such errors without further damaging the encoded quantum information.

Several methods have been proposed to detect and correct erasure errors with QECCs [52]. It is possible to correct erasure by deforming the original logical operator [53, 54], as well as by converting erasure errors into random Pauli errors by replacing the lost qubits with mixed states:

$$\frac{\mathbb{I}}{2} = \frac{1}{4}(\rho + X\rho X + Y\rho Y + Z\rho Z). \tag{2}$$

After replacing the qubits, one can perform stabilizer measurements as normally occurs in surface codes. Then, the erasure is converted into random Pauli errors with the exact probabilities (1/4) for $\{I, X, Y, Z\}$. This random Pauli can also be regarded as independent X and Z errors with a probability of 1/2. This allows for the decoding of an erasure error. The (surface code) peeling decoder [55], which is a linear-complexity erasure decoder using this procedure, has been proposed as a maximum-likelihood decoder for erasure errors in the surface code. We briefly overview the peeling decoder and its surface code generalization in Appendix D.

2.3 Applying quantum multiplexing to an error-corrected erasure channel

We describe the steps to perform error-corrected quantum communication over a multiplexed erasure channel using the surface code as an example, as illustrated in Fig. 2. For the first step, the sender prepares an encoded quantum state. Then, in the second step, the sender assigns and converts physical data qubits to photons. In the conventional case, different qubits are assigned to different photons, whereas when using quantum multiplexing, different qubits can be assigned to the same photon. In the instance of Fig. 2, qubits 0 and 6 are assigned to photon 0, qubits 1 and 7 are assigned to photon 1, etc. We will discuss the optimal assignment strategy later.

For the third step, a codeword passes through an optical channel which has the possibility of

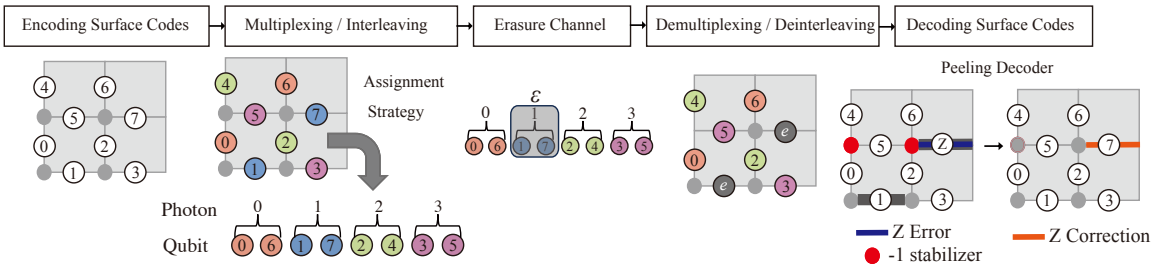


Figure 2: Flow of quantum communication with the surface code using multiplexed photons. In the first step, a quantum state is encoded into a surface code. Each circle with a number inside denotes the physical data qubits, and the grey circles without any numbers are auxiliary qubits used for stabilizer measurements. For the second step in a quantum multiplexing scenario, one assigns each physical qubit of the codeword to single photons using an assignment strategy. For instance, in this figure, two components of the time-bin DOF in each photon are used so that each photon can encode two qubits. There is freedom in which qubit is assigned to which photon, so it is necessary to define a function mapping qubits to photons. We call this function the assignment strategy. Here, the colors of the qubits indicate the photon to which it is assigned based on the assignment strategy. Then, the encoded photons pass through a lossy channel. Here, we assume that we know which photons have been lost during the transmission (erasure channel). If a photon has been lost, all the qubits assigned to this photon have been lost. Finally, we demultiplex the received photons and decode it to a codeword of the surface code using the peeling decoder [55] and a correction method for erasure errors as described in Sec. 2.2.

erasure errors. During the transmission, some photons can be lost, causing the loss of all qubits assigned to these photons as well. For instance, when photon 1 is lost, qubits 1 and 7 will also be lost as shown in Fig. 2, resulting in an erasure error. In the fourth step, the receiver reconstructs an incomplete codeword missing some qubits from the non-erased qubits assigned to the remaining photons. For the final step, the receiver replaces the missing qubits with mixed states as explained in Sec. 2.2. Effectively, this results in a codeword with random Pauli errors on some qubits. Next, the decoding algorithm estimates the errors, and the receiver performs a correction. In the second half of this paper, we examine the performance of this communication procedure via numerical simulation.

3 Multiplexed quantum communication with error-correcting codes

We propose three different scenarios in which multiplexing is used to enhance the efficiency of quantum communication. In each case, m qubits are encoded into each photon, and we compare them to the case of transmitting a codeword of a given code C without multiplexing ($m = 1$). The three scenarios are:

(A). m codewords of C are transmitted using the same number of photons as the $m = 1$ case;

(B). an m-times larger code from the same family as C is transmitted using the same number of photons as the $m = 1$ case;

(C). a codeword of C is transmitted using m-times fewer photons than the $m = 1$ case.

Examples for the case of the surface code are shown in Table. 1, where the parameters of this code are given as $[\![2d^2, 2, d]\!]$, with d being the code distance. Let us now explore each scenario in turn.

(A) Sending m different codewords

In the first scenario, the multiplexed photons are used to encode m codewords from m independent copies of the same code. The logical throughput of the channel increases m fold over the no-multiplexing case. One can assign qubits to photons so that each photon contains one qubit from a codeword of the distinct codes. The qubits from different codewords are correlated, but there is no correlation among the qubits in a fixed code. This correlation does not affect the logical error rate of the individual codes.

(B) Sending m-times bigger codewords

In the second scenario, a larger number of qubits is used to encode a single codeword from a code

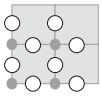
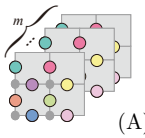
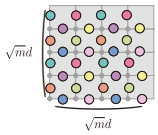
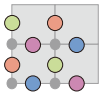
Scenarios	 without multiplexing	 (A)	 (B)	 (C)
Code parameters	$[\![2d^2, 2, d]\!]$	$[\![2d^2, 2, d]\!]$	$[\![2md^2, 2, \sqrt{m}d]\!]$	$[\![2d^2, 2, d]\!]$
Number of Codes	1	m	1	1
Number of Data Qubits	$2d^2$	$2md^2$	$2md^2$	$2d^2$
Number of Photons	$2d^2$	$2d^2$	$2d^2$	$\lfloor 2d^2/m \rfloor$
Logical Error Rate	-	Same as without quantum multiplexing	Affected by correlation	Affected by correlation

Table 1: Comparison of surface code communication without multiplexing and three scenarios with multiplexing. Red fonts show parameters that are improved by multiplexing. The case without multiplexing requires one qubit per photon. (A) The first scenario is only applicable when sending multiple codewords. This enables one to send more codewords with the same number of photons, drastically improving the channel's throughput. (B) The second scenario sends the same number of codewords but uses a larger code, thus improving the error tolerance. (C) The third scenario sends the same codeword with fewer photons, which also significantly improves the channel's throughput. The number of photons required in scenario (C) is $\lfloor 2d^2/m \rfloor$, where $\lfloor x \rfloor$ is the floor function of x.

in the same family with an m-fold longer length. If this scenario is applied to the surface code, it achieves $\sqrt{m}$ times larger distance than the no-multiplexing case ($m = 1$).

Fig. 3 shows a Monte Carlo simulation of the logical Z error rate for this scenario for the surface code. We use surface codes only on a $2D$-torus (i.e. toric codes) in this paper. To determine whether a logical Z error occurred, we checked whether the errors left after the decoding process were anti-commutative with a logical X operator of any logical qubit in the codeword. As logical Y errors can be viewed as combinations of logical X and Z errors, we also count these as logical Z errors for the purpose of these plots. Each data point in the simulation is obtained from 10^5 shots, and the error bar is given by the Agresti–Coull interval [56]. For simplicity, we assign qubits to multiplexed photons uniformly at random in this section. This scenario introduces correlations in errors between the qubits in the code, which may degrade the performance. However, if m is sufficiently small relative to the code size, the benefit gained by increasing the code size is more significant. All the programs we used to simulate multiplexed quantum communication with surface codes are available here [57]. The logical error rate significantly decreases as the code size and m increase.

Readers may wonder why the logical Z error rate converges to 0.75, even though for a loss probability of 1 the code will be completely erased. This is due to the fact that, following such a complete erasure, the recovery procedure will reinitialize the code in a random state subject to one of the logical operators II, ZI, IZ, ZZ with equal probability, and so 25% of the time we do not observe a logical Z error. Importantly, this finite recovery probability is observed because, in this case, we care only about the presence/absence of logical Z operators, and it does not imply that an arbitrary quantum state can be recovered.

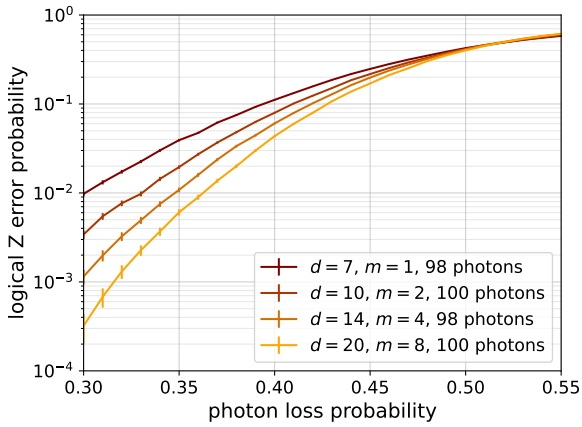


Figure 3: Performance of $[\![2d^2, 2, d]\!]$ surface codes in scenario (B) with about 100 photons. Each curve shows the case with different code sizes and the number of qubits encoded in each photon. The logical error rate can be reduced by increasing the number of qubits per photon m and the code distance d.

(C) Sending original codewords with fewer photons

In the third scenario, a smaller number of photons are used to encode a single codeword. The code parameters are the same as the case without multiplexing. It has no restriction on the number of codewords and can improve the efficiency of surface code communication in general. This method introduces a correlation to the errors. Fig. 4 shows this scenario's logical Z error rate versus the photon loss probability for different values of m. It shows that as m increases, the logical error rate also increases.

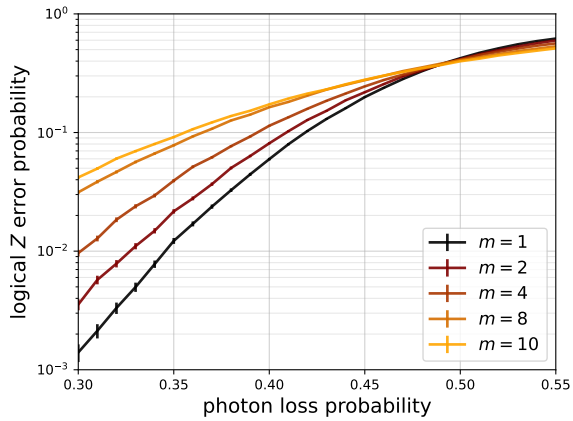


Figure 4: Scenario (C) multiplexing performance for the $[\![200, 2, 10]\!]$ surface code with multiplexing using different values of m (the number of qubits per photon). The assignment of qubits to photons is uniformly random. Increasing m allows code words to be transmitted with fewer photons, but the logical error rate increases because multiple qubits in the same photon have strongly correlated errors.

Although the number of photons is reduced compared to the no-multiplexing case, the effects of the correlated errors can be very detrimental to the performance of such a system. Ideally, these detrimental effects can be reduced by strategically assigning qubits to the multiplexed photons. In the next subsection, we explore five different strategies for qubit assignment.

3.1 Limitation and fault-tolerance of multiplexed communication

Generating time-bin multiplexed photons requires linear optical elements such as polarizing beam splitters, delay lines, wave plates, and optical switches, as shown in Fig. 1 (A). In particular, the latter suffer from loss errors that are detrimental to the overall performance of the system. In [29], the authors analyze the impact of imperfect optical switches on the performance of a purification protocol. They show that, for an optical switch efficiency equal to 99%, adding one time-bin mode to a polarized photon only slightly reduces the performance of the system. However, when several optical switches are in use to encode multiple time-bin modes, much higher efficiencies are required. Although not fully achieved with the current devices, future high-efficiency optical switches are required for several quantum technologies.

While it is possible for multiplexing to lead to correlated errors between qubits stored in the same photon, we do not expect this to compromise fault tolerance except in the case where these qubits correspond to the support of a logical operator of the code, and the majority of our assignment strategies are designed to prevent this. More generally, if we require that the number of qubits per photon is a) independent of the size of the code and b) smaller than the code distance, then these correlated errors can, at most, result in a constant-factor reduction of the distance, and so long as the non-multiplexed procedure is fault-tolerant, the multiplexed procedure will be fault-tolerant as well.

4 Quantum Communication with Multiplexed Surface Codes

4.1 Assignment Strategies for Surface Codes

In this section, we describe five strategies for assigning qubits to photons which take advantage of multiplexing, and then evaluate the impact of these strategies on communication performance. These strategies assume that each photon contains a fixed number of qubits m and can be applied in both scenarios **(B)** and **(C)**. Because **(B)** can be regarded as a scaled version of scenario **(C)**, we will focus specifically on surface code communication using the latter case, where we send the original code word with $\lfloor 2d^2/m \rfloor$ photons.

Strategy i and ii: pair with minimum and maximum distance Strategies i and ii are applicable to the case with $m = 2$. Strategy i assigns the

nearest neighbor pair of qubits to the same photon, which forms an L-shape in the 2D lattice of the surface code as shown in the example of Fig. 5(A). This minimizes the Manhattan distance in the lattice between the qubits in the same photon. Strategy ii assigns the qubit at coordinates (i, j) on the lattice and the qubit at coordinates $(i+d/2-1 \bmod d, j+d/2-1 \bmod d)$ to the same photon. An example is shown in Fig. 5(B). This is the arrangement that maximizes the Manhattan distance, in contrast to strategy i.

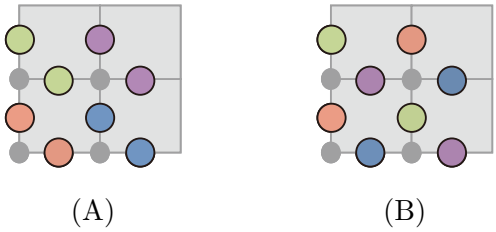


Figure 5: Examples showing possible assignments of qubits to photons. Each colored circle denotes a qubit, and the color indicates the photon to which the qubit is assigned. Each non-colored (gray) small circle denotes auxiliary qubits for stabilizer measurement. Strategy i (shown in (A)) minimizes the distance between qubits in the same photon, while strategy ii (shown in (B)) maximizes this distance. Note that this code is defined on the torus represented as a lattice with periodic boundary conditions.

Strategy iii: random Strategy iii is a method in which qubits are selected uniformly at random and assigned to photons.

Strategy iv: random + threshold Strategy iv is designed to increase the separation between qubits assigned to the same photon while exploiting randomness. The strategy works by randomly selecting qubits and accepting them as the set for a photon only if the distance is greater than a certain threshold. If no suitable set of qubits can be found, then the threshold value is reduced.

Strategy v: stabilizer Strategy v assigns the qubit support of stabilizer generators to the same photon. Realizations of this assignment strategy on a 4×4 surface code are shown in Fig. 6.

We describe the details and the motivation of each strategy in Appendix C.

4.2 Performance of the assignment strategies

Here, we show the performances of these strategies observed in numerical simulations.

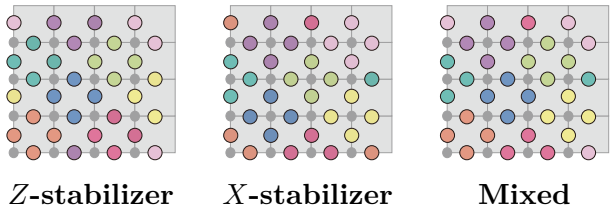


Figure 6: Examples of the stabilizer-based photon assignment strategy for a surface code on a 4×4 lattice. Edges representing qubits in the lattice are marked with colored nodes indicating photon assignment. In this lattice picture, the qubit support of Z-type stabilizer generators corresponds to squares, and of X-type stabilizer generators corresponds to crosses. Each photon in the stabilizer assignment strategy represents the qubit-support of one of these stabilizers.

Fig. 7 (A) shows the performance of strategies i through iv, which are applicable to the case of $m = 2$. The distance-maximizing strategy (grey) outperforms the distance-minimizing strategy (brown). Logical errors in the surface code correspond to errors covering a longitude or meridian curve on the torus (a vertical or horizontal closed loop in the periodic lattice). When decoding erasure errors, logical errors can only occur when the qubit-support of one of these vertical or horizontal loops is entirely erased. When adjacent qubits in the lattice are erased, as in the case with the distance-minimizing photon assignment strategy, clusters of errors are more likely to cover such loops in the torus. Hence, it is not surprising that the distance-maximizing strategy outperforms the distance-minimizing strategy in our numerical simulations. We also see that the strategies with randomness (iii and iv) outperform the deterministic ones (i and ii). In particular, strategy iv outperformed the other strategies, although there was an increase in logical Z-error probability compared to no multiplexing.

Next, we compare the logical Z error rates of strategies iii, iv, and v with $m = 4$ in Fig. 7 (B). Assignment strategies based on one type of stabilizer create a bias in observed logical error rates. Strategy v can be generalized to any stabilizer code, and the assignment strategy based only on the support of X or Z stabilizers will increase the error rate of one of X or Z and decrease the other. This result implies that stabilizer-based assignment may be useful in quantum error correction codes with different X and Z distances.

Both Fig. 7 (A) and (B) showed that the strat-

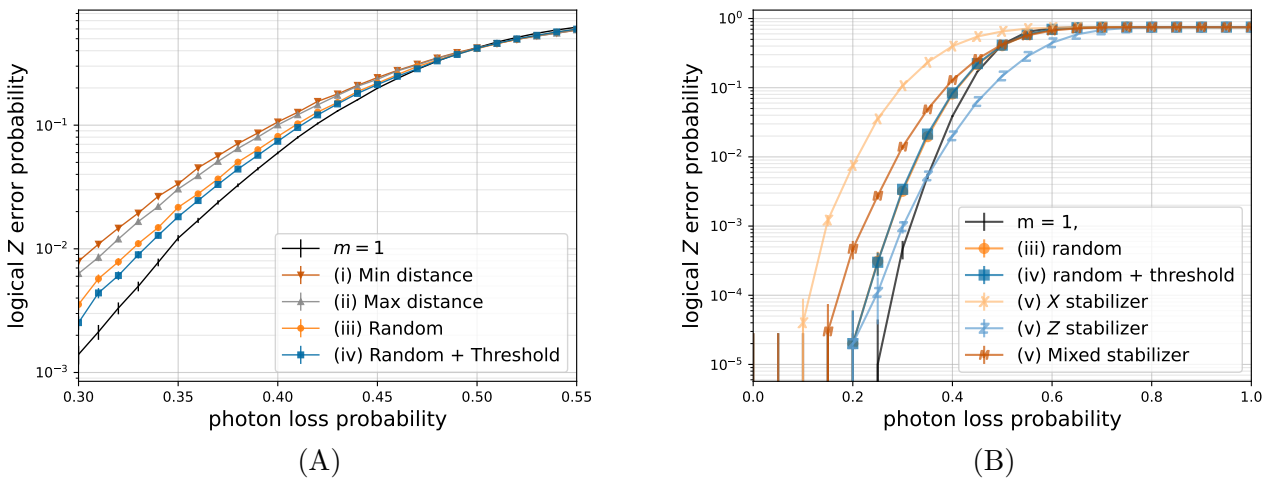


Figure 7: Comparison of multiplexing photon-assignment strategies for surface codes. Logical Z error rate versus photon loss probability. The black curve shows the case without multiplexing. (A) The code parameters are $[\![200, 2, 10]\!]$ and $m = 2$. The gray/brown curve shows the case for the assignment strategy for minimizing (strategy i) / maximizing (strategy ii) the distance between a pair of qubits in the same photon. The orange curve shows the case for uniform randomness (strategy iii), and the blue line shows random + threshold (strategy iv), based on Algorithm 1. (B) The code parameters are $[\![288, 2, 12]\!]$ and $m = 4$. The Z stabilizer-based assignment (light blue curve) outperformed the X stabilizer-based assignment (light orange curve) for logical Z errors. The mixed stabilizer-based assignment strategy performs between X and Z. Strategy iii (orange circle) and iv (blue square) outperform the other assignment strategies when the photon loss probability is low.

egy iv random + threshold outperformed the other strategies. Maximizing the distance between qubits while also introducing randomness gives the largest boost in performance against the occurrence of logical errors. Note that no assignment strategy does better than the case with $m = 1$ where no multiplexing is used.

We also analyzed the difference in the performance between cases with multiplexing ($m = 4$) and without it ($m = 1$), as shown in Fig. 8. When physical error rates are low, this difference decreases with increasing code distance, suggesting that the downsides of multiplexing are less significant in larger codes.

To assess the impact of multiplexing on the logical error rate in more realistic use cases, we evaluated the performance in a mixed error model where both Pauli errors and photon losses are present. We performed a simulation with the following flow.
(1) Random Pauli errors on sender's processor $\xrightarrow{\text{transfer out and send}}$ (2) Erasure errors on (multiplexed) photonic states $\xrightarrow{\text{transfer in}}$ (3) Random Pauli errors on receiver's processor $\rightarrow$ (4) Quantum nondemolition measurement to detect erasure errors $\rightarrow$ (5) Replace erased qubits with mixed state $\rightarrow$ (6) Stabilizer measurement which maps erasure errors to random Pauli errors $\rightarrow$ (7) Decoding procedure. Here, "transfer out" is a procedure to encode qubits into a photon, and "transfer in" is the opposite procedure.

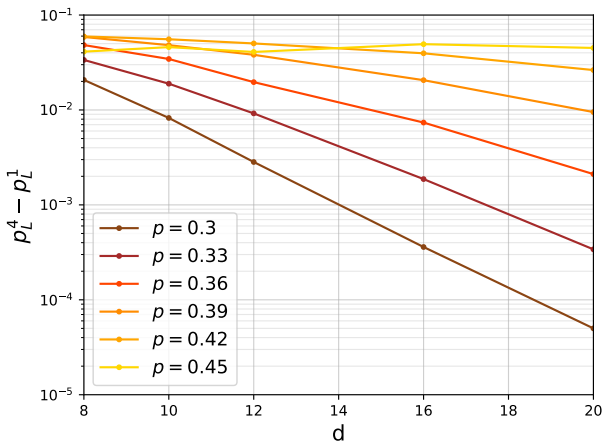


Figure 8: Difference of logical Z error rates for $m = 4$ (p_L^4) and $m = 1$ (p_L^1) for various photon loss probabilities (p). For low p ($0.3 \sim 0.42$), the gap decreases to 0 as d increases.

Note that random Pauli errors in (1) have a smaller effect on the logical error rate than

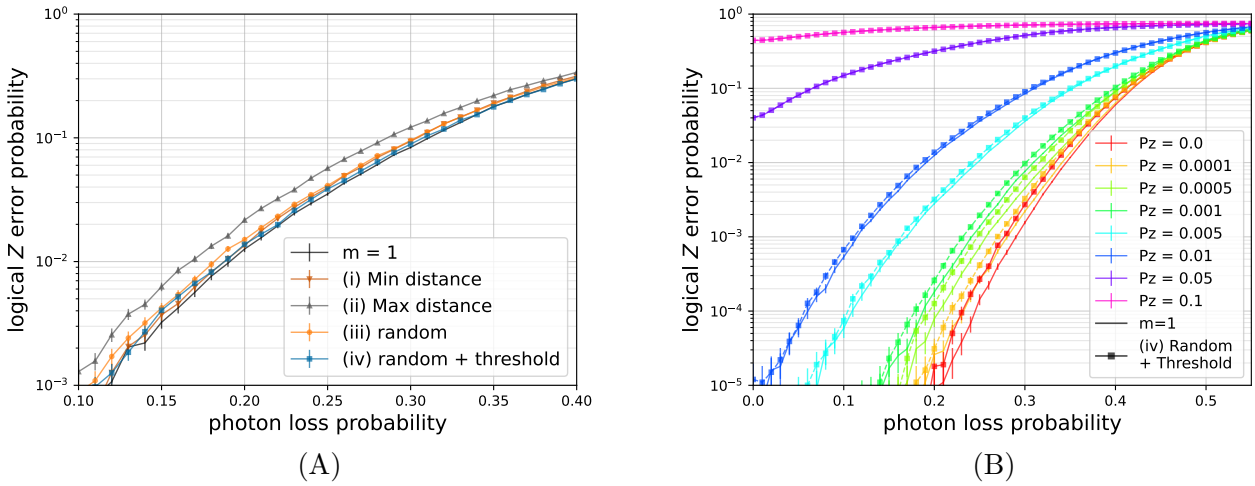


Figure 9: (A) Comparison of $m = 2$ photon-assignment strategies on the $[\![200, 2, 10]\!]$ surface code with the combined error model. The combined error consists of erasure errors and 1% Pauli Z errors occurring before and after the multiplexed communication. (B) The logical error rate for $m = 1$ and $m = 2$ (strategy (iv)) with different probability of Pauli Z errors P_Z. 10^6 samples are taken. Color indicates P_Z, and solid and dashed lines indicate $m = 1$ and $m = 2$, respectively.

in (3) because some erasure errors in (2) will eliminate the effect of (1) on the erased qubits. In this setting, there are Pauli errors only before and after the communication part, and there are no Pauli errors on the multiplexed channel. The behavior of Pauli errors on the multiplexed state is not as obvious as the single-qubit Pauli errors because, for such a case, the Pauli errors may be on qudits, and it can have a larger effect than single-qubit Pauli errors. This qudit Pauli on the multiplexed channel is complex, so we will leave this problem to future work.

Note that we have implemented and used the Union-find decoder [58] for this new simulation, which can handle the combined errors. Fig. 9 (A) shows the logical error rate with the combined error for different strategies. The performance of strategies (i) and (ii) is reversed compared to the case without Pauli error in Fig. 7 (A). The strategy (iv) achieves a logical error rate relatively closer to $m = 1$ than (iii) does. Fig. 9 (B) shows the logical error rate versus photon loss probability with different Pauli error rates P_Z for $m = 1$ and $m = 2$ (strategy (iv)). The $m = 1$ case outperforms strategy (iv) for low values of P_Z, but for high values, strategy (iv) is close to no multiplexing and outperforms only slightly for some loss error rates. These indicate that the method of choosing an appropriate assignment strategy in more practical cases may require further consideration.

Note that the threshold [53] for the surface code with Union-find decoder is 9.9 % [58], but here we have random Pauli Z errors occurring before and after the communication process, and so we expect the threshold to be roughly half the original threshold.

5 Quantum Communication with Multiplexed Hypergraph Product Codes

5.1 Hypergraph Product Code Structure

In addition to our exploration of the surface code, we also consider the use of multiplexing with hypergraph product (HGP) codes [59], of which surface codes are a special case. HGP codes are a special class of CSS code defined using any two classical linear codes. They are of particular interest because they can have an asymptotically finite rate as the code length increases (in contrast with the surface code, which has a rate approaching 0) and distance proportional to the minimum distance of the classical codes; in the best case, this is proportional to the square root of the quantum code length. They are also considered practical candidates for FTQC codes.

Given classical parity check matrices H_1 and H_2 with sizes $r_1 \times n_1$ and $r_2 \times n_2$, respectively, we may define the matrices H_X and H_Z of a CSS code via the formulas

$$H_X = (H_1 \otimes I_{n_2} | I_{r_1} \otimes H_2^T) \quad (3)$$
$$H_Z = (I_{n_1} \otimes H_2 | H_1^T \otimes I_{r_2}), \quad (4)$$

where I_{n_1}, I_{n_2}, I_{r_1}, I_{r_2} are identity matrices with sizes n_1, n_2, r_1, and r_2, respectively. These matrices satisfy the condition $H_X H_Z^T = 0$ by construction and hence define a valid CSS code $\mathrm{HGP}(H_1, H_2)$. When H_1 and H_2 are low-density parity checks (LDPC), H_X and H_Z will also be LDPC. The sizes of H_X and H_Z are determined by the sizes of the input classical matrices according to the formulas

$$H_X = [r_1 n_2 \times (n_1 n_2 + r_1 r_2)] \quad (5)$$
$$H_Z = [r_2 n_1 \times (n_1 n_2 + r_1 r_2)]. \quad (6)$$

These both simplify to $rn \times (n^2 + r^2)$ in the special case where $r_1 = r_2 = r$ and $n_1 = n_2 = n$.

HGP codes have a geometrically rich Tanner graph structure which can be visualized as the cartesian product of the Tanner graphs for the two input classical codes as shown in Fig. 10. The subgraph corresponding to each row and column in this Tanner graph block structure can be understood as the classical Tanner graph for one of the classical codes used in the construction. As shown in the figure, qubits are represented by circular nodes, and stabilizer checks of both types

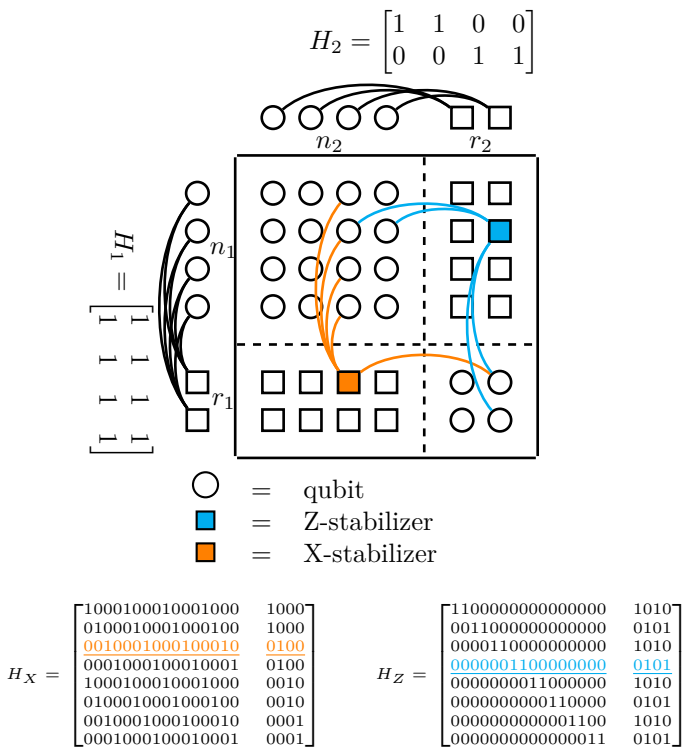


Figure 10: Example of the Tanner graph for a simple HGP code $\mathrm{HGP}(H_1, H_2)$ constructed from two classical codes with parity check matrices H_1 and H_2. This is the cartesian product of two classical Tanner graphs, and the subgraph corresponding to each row and column in the product is a copy of one of these classical Tanner graphs. This product structure can be partioned into four quadrants, each representing a different structural component of the HGP code. The nodes in the upper-left and lower-right blocks denote qubits. The nodes in the upper-right block denote Z-stabilizer generators; these correspond to the rows of H_Z. Similarly, the nodes in the lower-left block denote X-stabilizer generators; these correspond to the rows of H_X.

are represented by square nodes. Additional details regarding this construction are discussed in Appendix E.1.

Surface codes may also be recovered as a special case of hypergraph product code. Using parity check matrices H_1 and H_2 for a classical repetition code, $\mathrm{HGP}(H_1, H_2)$ is exactly the toric surface code. Hence, adapting the multiplexing strategies discussed in Sec. 4 to this more general class of codes is a natural next question. However, the linear-time maximum-likelihood generalization of the peeling decoder [55] used in our previous simulations is only defined for the special case of the surface code. This decoder leverages the lattice structure of the surface code to en-

sure that the erasure subgraph can be completely peeled (Appendix D.3), but this technique does not apply to generic HGP codes. Instead, we introduce another generalization of the peeling algorithm to extend our numerical analysis to HGP codes as well.

5.2 Pruned-Peeling + VH Decoder

The *pruned peeling + VH decoder* [60] is a generalization of the peeling algorithm (outlined in Appendix D) specifically designed for HGP codes, where VH stands for Vertical-Horizontal. It has quadratic complexity and close to maximum-likelihood performance at low erasure rate, making it practically useful for our simulations. This decoder is a modified version of the standard classical peeling decoder based on analysis and correction of two common types of *stopping sets*, which are patterns of erased qubits that cannot be corrected by simple peeling.

A *stabilizer stopping set* occurs when the erasure pattern covers the qubit support of an X- or Z-type stabilizer. The *pruned peeling decoder* attempts to fix these by removing a qubit from the erasure, thus "breaking" the stabilizer support and possibly allowing the peeling algorithm to become unstuck. *Classical stopping sets* are patterns of erased qubits supported entirely on a single row or column in the HGP Tanner graph block structure of Fig. 10; any stopping set for a HGP code can be decomposed into a union of components of this form. The *VH decoder* algorithm attempts to order and efficiently solve each of these classical stopping sets in sequence. The name "vertical-horizontal" refers to the fact that, in the HGP Tanner graph picture, individual classical stopping sets are either entirely supported on a column (vertical) or a row (horizontal). The combination of these decoding strategies is referred to as the *combined decoder* (peeling + pruned peeling + VH). A more detailed explanation is included in Appendix D.4.

The combined decoder is not a maximum likelihood decoder. In addition to logical errors, there still exist patterns of erased qubits where the decoder becomes stuck in a stopping set, leading to a decoder failure. We introduce the term *error recovery failure* to refer to both decoder failures and logical errors (discussed in Appendix E.2). Our numerical simulations for HGP codes are always with respect to the *error recovery failure rate*.

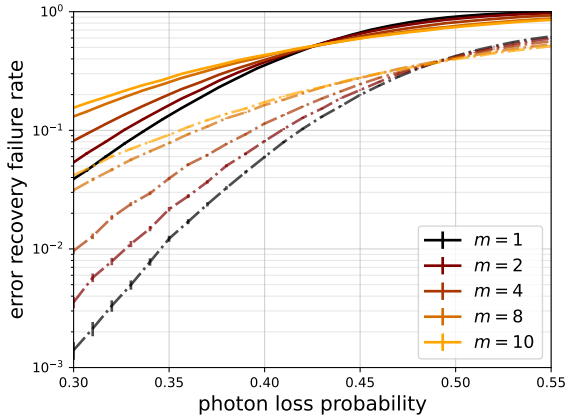
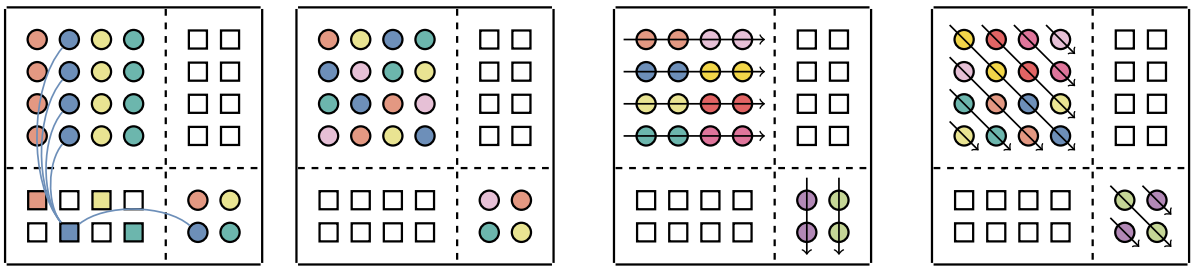


Figure 11: Performance of non-ML *combined decoder* (peeling + pruned peeling + VH) as shown in solid curves and the *surface code peeling decoder* as shown in dashed curves applied to the $[\![200, 2, 10]\!]$ surface code using the uniformly random assignment strategy with different numbers of qubits in a single photon, m.

Fig. 11 shows the numerical performance of the combined decoder applied to the 10×10 surface code for comparison with the replotted data from Fig. 4, which shows the performance of the ML decoder applied to this same code. The performance degradation is explained by the presence of decoder failures that do not exist in the ML case. Even though it is not ML, the pruned peeling + VH decoder is still practically useful for our numerical simulations since decoder failures are infrequent at low erasure rates.

5.3 Assignment Strategies for HGP Codes

As with the surface code, quantum multiplexing can also be utilized with HGP codes. In this section, we analyze the performance of HGP code communication in scenario **(C)**. The scenarios previously proposed in Sec. 4 are also valid for HGP codes, but unlike the special case of the surface code, the distance between any two qubits in a generic HGP code is not easily inferred from a grid. Hence, we do not consider the previously introduced strategies which use distance. We also introduce several new strategies for HGP codes based on stopping sets for the pruned peeling + VH decoder. These strategies are summarized in Table 2, and technical details are included in Appendix E.3.



Strategy ii. Stabilizer	Strategy iii. Sudoku	Strategy iv. Row-Column	Strategy v. Diagonal

Table 2: Examples of four different photon assignment strategies for the simple HGP code shown in Fig. 10. **(ii.)** Each photon in the *stabilizer strategy* is the qubit-support of an X or Z-type stabilizer generator, identified as a row of H_X or H_Z. The number of qubits per photon is a fraction or multiple of the weight of the corresponding row. **(iii.)** In the *sudoku strategy*, each qubit of a given photon is contained in a different row or column of the HGP Tanner graph. **(iv.)** Using the *row-column strategy*, each qubit of a given photon is contained in the same row or column of the HGP Tanner graph. **(v.)** Photons from the *diagonal strategy* contain qubits from the same diagonal slice of the HGP Tanner graph, allowing diagonal lines to wrap around. For strategies iii., iv., and v., the number of qubits per photon is a fraction or multiple of the shortest side length in the block structure.

Strategy i: random The simplest assignment strategy is based on assigning qubits to photons uniformly at random.

Strategy ii: stabilizer The stabilizer assignment strategy assigns qubits to photons so that photons correspond to the qubit-support of a stabilizer. This strategy is motivated by the fact that the pruned peeling decoder is designed to correct erased stabilizers.

Strategy iii: sudoku The sudoku strategy assigns qubits to photons at random subject to the condition that qubits within a given photon come from different rows and columns in the HGP Tanner graph structure. It is motivated by the goal of reducing classical stopping sets, a common source of peeling decoder failures for HGP codes. We name this the sudoku strategy due to its resemblance to the popular game.

Strategy iv: row-column In contrast to sudoku, the row-column strategy chooses qubits in a given photon from the same row or column of the HGP Tanner graph structure. It seeks to maximize the number of classical stopping sets and hence decoder failures. The row-column strategy can be interpreted as a worst-case scenario.

Strategy v: diagonal The diagonal assignment strategy is based on dividing the qubit blocks in the HGP Tanner graph into diagonal slices. Qubits from the same diagonal slice are assigned to the same photon. This is a modified version of the sudoku strategy which does not use randomness but still seeks to minimize classical stopping sets and hence decoder failures.

To compare the effectiveness of these strategies, we have simulated their performance for several codes at different multiplexing values as shown in Fig. 12 and Fig. 13. To understand these results, the case with no-multiplexing ($m = 1$) is used as the baseline. An assignment strategy is considered good if its failure rate is not significantly worse than the $m = 1$ case. Interestingly, our numerical simulations consistently show that the performances of some strategies (random, sudoku, and diagonal) are almost equivalent to or even exceed the $m = 1$ case, even at high multiplexing values. However, the row-col and stabilizer strategies are never seen to be effective in our results.

Fig. 12 shows an example of a code where the diagonal strategy consistently outperforms all other strategies, even the no-multiplexing case, and even at low erasure rates. This result is significant because even though multiplexing reduces the number of required physical resources, it is possible to improve the decoding performance while doing so. In fact, an analysis of these results reveals that the diagonal strategy yields fewer logical errors than the no-multiplexing case at the same physical erasure rate. This appears to be a feature of the structure of the logical operators in the randomly generated code used in this simulation, even though the strategy was not designed with this in mind. This also explains the gap between the sudoku and diagonal strategies, both of which have similar amounts of decoder failures but differ with respect to logical errors.

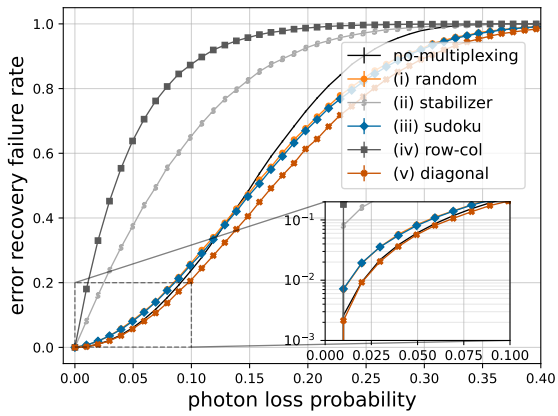


Figure 12: Multiplexing decoder performance for a ⟦320,82⟧ non-equal block (16×16 and 8×8) HGP code at fixed $m = 8$. In this example, strategy (v) diagonal outperforms all other strategies, including the no-multiplexing case.

These results show that strategies designed to avoid decoder failures can have comparable (or even favorable) performance relative to the no-multiplexing case.

Although not identical, we observe similar performance for a larger HGP code, as shown in Fig. 13. The random, sudoku, and diagonal strategies have nearly identical performance to the no-multiplexing case regardless of the chosen multiplexing number. (Simulations include $m \in \{2, 4, 8, 16\}$, although only plots for $m = 4$ and $m = 16$ are shown.) Furthermore, these results hold consistently at a low erasure rate, which is the regime of practical interest. This is significant because it implies there is no loss in performance when multiplexing, even though fewer physical resources are required, provided the assignment strategy is adapted to the decoder. If an ML decoder were used (e.g., Gaussian elimination rather than peeling + pruned peeling + VH), a gap is expected between the multiplexing and no-multiplexing cases. However, given that the combined decoder is a faster, more efficient alternative to a true ML decoder for HGP codes, these results are very promising.

All the programs we used to simulate multiplexed quantum communication with HGP codes are available here [61].

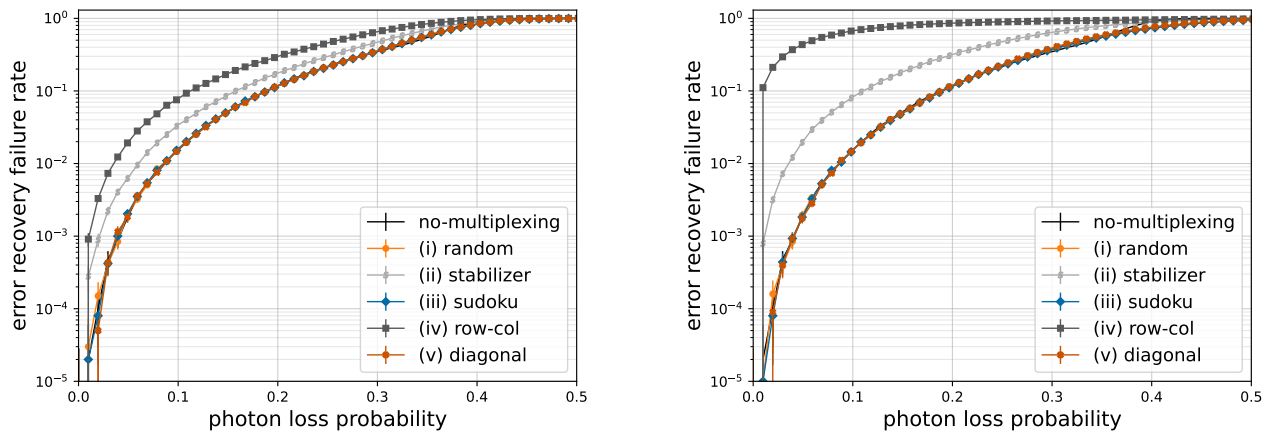
6 Discussion and Conclusion

We proposed three error-corrected quantum information processing scenarios for quantum memory storage and communication with quantum multiplexing over an erasure channel. We have shown that quantum multiplexing can improve throughput or resilience to errors, easing the bottleneck in quantum systems. This work can be adapted to error-corrected quantum communication [28] with quantum interconnects, quantum repeaters, and multimode quantum memory [62].

For multiplexed quantum communication, if multiple qubits in a single code word are encoded into the same photon, a correlation of errors in those qubits will be introduced. The simulation results show that it leads to an increase in the logical error rate. We showed that this performance gap can be significantly mitigated by introducing a code-aware (or decoder-aware) strategy to assign qubits to photons, which exploits code structure. For surface codes in particular, both randomness and distance maximization are important factors for achieving this. For HGP codes with the VH decoder, minimizing decoder failures was found to be the most important factor.

These techniques can also be exploited to benefit other families of codes and decoders. Furthermore, it is possible to deal with the gap between the multiplexing and no-multiplexing cases by increasing the code size. We have also shown that it is possible to introduce biased error by using a stabilizer-based assignment strategy. In the special case of the diagonal strategy for the HGP code of Fig. 12, we see that the photon-correlated errors offer an improvement over the no-multiplexing case. In this example, the improvement can be explained by the fact that the diagonal strategy reduces logical errors in addition to decoder failures. Furthermore, this shows the existence of strategies that improve over no-multiplexing despite the fact that fewer resources are used.

Even though a linear-time ML decoder has not yet been discovered for generic HGP codes as it has been for surface codes, the use of HGP codes with quantum multiplexing should not be overlooked. Unlike surface codes, which have fixed dimension 2, HGP codes can be chosen so that code dimension k increases linearly with code length n. This can be significant for applications us-



(A) $m = 4$ (B) $m = 16$

Figure 13: Comparisons of multiplexing decoder performance for a ⟦512,8⟧ equal-block (16×16) HGP code obtained from the symmetric construction with $r = n = 16$ using various assignment strategies for $m = 4$ and $m = 16$. In both cases, the random, sudoku, and diagonal strategies are seen to be effectively equivalent to the no-multiplexing case, even at low erasure rates.

ing increasingly long codes since the code rate need not approach 0 in the HGP case. Furthermore, while an ML decoder is ideal, non-ML decoders are often good enough for error correction in the regime of practical interest. The speed-up gained by using a more efficient decoder can offset new errors that might arise when using a slower decoder. Our numerical results for HGP codes show that decoder-aware strategies enable us to gain all of the benefits of quantum multiplexing without sacrificing any additional performance. This, combined with the throughput advantage HGP codes have over the surface code, is a very promising practical result.

Although we propose several promising candidates, the optimal assignment strategy for both surface codes and HGP codes is still unknown. Furthermore, in actual communication with quantum multiplexing, various errors may occur when converting qubits in the quantum processor to photons, measuring stabilizers, and substituting erased qubits with mixed states. How to deal with these errors is a practically important next question.

Multiplexing could also be used for qubit → qudit encodings in non-photonic systems where loss errors are not the dominant source of noise. In these cases error locations may not be known and so knowledge of the assignment strategy could be used to inform decoding.

Acknowledgements

SN acknowledges Dan Browne, Antonio deMarti iOlius, and Hon Wai Lau for valuable discussions throughout this project. This work was supported by JSPS KAKENHI Grant Number JP22KJ1436, JP21H04880, the MEXT Quantum Leap Flagship Program (MEXT Q-LEAP) Grant Number JPMXS0118069605, the JST Moonshot R&D Grant Number JPMJMS2061 and JPMJMS226C, and a travel budget from the National Institute of Informatics.

References

[1] P.W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In Proceedings 35th Annual Symposium on Foundations of Computer Science. Pages 124–134. (1994).

[2] Lov K Grover. "A fast quantum mechanical algorithm for database search". In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. Pages 212–219. (1996).

[3] Thomas Häner, Martin Roetteler, and Krysta M. Svore. "Factoring using 2n + 2 qubits with toffoli based modular multiplication". Quantum Info. Comput. **17**, 673–684 (2017).

[4] Nobuyuki Yoshioka, Tsuyoshi Okubo, Yasunari Suzuki, Yuki Koizumi, and Wataru Mizukami. "Hunting for quantum-classical crossover in condensed matter problems". npj Quantum Information **10**, 45 (2024).

[5] Sebastian Krinner, Simon Storz, Philipp Kurpiers, Paul Magnard, Johannes Heinsoo, Raphael Keller, Janis Luetolf, Christopher Eichler, and Andreas Wallraff. "Engineering cryogenic setups for 100-qubit scale superconducting circuit systems". EPJ Quantum Technology **6**, 2 (2019).

[6] Shuhei Tamate, Yutaka Tabuchi, and Yasunobu Nakamura. "Toward realization of scalable packaging and wiring for large-scale superconducting quantum computers". IEICE Transactions on Electronics **105**, 290–295 (2022).

[7] David Awschalom, Karl K Berggren, Hannes Bernien, Sunil Bhave, Lincoln D Carr, Paul Davids, Sophia E Economou, Dirk Englund, Andrei Faraon, Martin Fejer, et al. "Development of quantum interconnects (quics) for next-generation information technologies". PRX Quantum **2**, 017002 (2021).

[8] Christopher Monroe, Robert Raussendorf, Alex Ruthven, Kenneth R Brown, Peter Maunz, L-M Duan, and Jungsang Kim. "Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects". Physical Review A **89**, 022317 (2014).

[9] Koji Azuma, Sophia E Economou, David Elkouss, Paul Hilaire, Liang Jiang, Hoi-Kwong Lo, and Ilan Tzitrin. "Quantum repeaters: From quantum networks to the quantum internet". Reviews of Modern Physics **95**, 045006 (2023).

[10] WJ Munro, R Van Meter, Sebastien GR Louis, and Kae Nemoto. "High-bandwidth hybrid quantum repeater". Physical review letters **101**, 040502 (2008).

[11] Liang Jiang, Jacob M Taylor, Kae Nemoto, William J Munro, Rodney Van Meter, and Mikhail D Lukin. "Quantum repeater with encoding". Physical Review A **79**, 032325 (2009).

[12] William J Munro, Ashley M Stephens, Simon J Devitt, Keith A Harrison, and Kae Nemoto. "Quantum communication without the necessity of quantum memories". Nature Photonics **6**, 777–781 (2012).

[13] Rodney Van Meter. "Quantum networking". John Wiley & Sons. (2014).

[14] Sreraman Muralidharan, Linshu Li, Jungsang Kim, Norbert Lütkenhaus, Mikhail D Lukin, and Liang Jiang. "Optimal architectures for long distance quantum communication". Scientific reports **6**, 20463 (2016).

[15] William J Munro, Nicolo'Lo Piparo, Josephine Dias, Michael Hanks, and Kae Nemoto. "Designing tomorrow's quantum internet". AVS Quantum Science**4** (2022).

[16] Jianwei Wang, Damien Bonneau, Matteo Villa, Joshua W Silverstone, Raffaele Santagati, Shigehito Miki, Taro Yamashita, Mikio Fujiwara, Masahide Sasaki, Hirotaka Terai, et al. "Chip-to-chip quantum photonic interconnect by path-polarization interconversion". Optica **3**, 407–413 (2016).

[17] Alexander I Lvovsky, Barry C Sanders, and Wolfgang Tittel. "Optical quantum memory". Nature photonics **3**, 706–714 (2009).

[18] Y-W Cho, GT Campbell, JL Everett, J Bernu, DB Higginbottom, MT Cao, J Geng, NP Robins, PK Lam, and BC Buchler. "Highly efficient optical quantum memory with long coherence time in cold atoms". Optica **3**, 100–107 (2016).

[19] A Robert Calderbank and Peter W Shor. "Good quantum error-correcting codes exist". Physical Review A **54**, 1098 (1996).

[20] Daniel Gottesman. "Stabilizer codes and quantum error correction" (1997).

[21] Daniel Gottesman. "Fault-tolerant quantum computation with higher-dimensional systems". In NASA International Conference on Quantum Computing and Quantum Communications. Pages 302–313. Springer (1998).

[22] A Yu Kitaev. "Fault-tolerant quantum computation by anyons". Annals of physics **303**, 2–30 (2003).

[23] Shota Nagayama, Byung-Soo Choi, Simon Devitt, Shigeya Suzuki, and Rodney Van Meter. "Interoperability in encoded

quantum repeater networks". Physical Review A **93**, 042338 (2016).

[24] Sergey B Bravyi and A Yu Kitaev. "Quantum codes on a lattice with boundary" (1998).

[25] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. "Surface codes: Towards practical large-scale quantum computation". Physical Review A **86**, 032324 (2012).

[26] Clare Horsman, Austin G Fowler, Simon Devitt, and Rodney Van Meter. "Surface code quantum computing by lattice surgery". New Journal of Physics **14**, 123011 (2012).

[27] Markus Grassl, Willi Geiselmann, and Thomas Beth. "Quantum reed—solomon codes". In Applied Algebra, Algebraic Algorithms and Error-Correcting Codes: 13th International Symposium, AAECC-13 Honolulu, Hawaii, USA, November 15–19, 1999 Proceedings 13. Pages 231–244. Springer (1999).

[28] Austin G Fowler, David S Wang, Charles D Hill, Thaddeus D Ladd, Rodney Van Meter, and Lloyd CL Hollenberg. "Surface code quantum communication". Physical review letters **104**, 180503 (2010).

[29] Nicolo Lo Piparo, William J Munro, and Kae Nemoto. "Quantum multiplexing". Physical Review A **99**, 022337 (2019).

[30] Shin Nishio, Nicolò Lo Piparo, Michael Hanks, William John Munro, and Kae Nemoto. "Resource reduction in multiplexed high-dimensional quantum reed-solomon codes". Physical Review A **107**, 032620 (2023).

[31] Nicolo Lo Piparo, Michael Hanks, Claude Gravel, Kae Nemoto, and William J Munro. "Resource reduction for distributed quantum information processing using quantum multiplexed photons". Physical Review Letters **124**, 210503 (2020).

[32] Nicolo Lo Piparo, Michael Hanks, Kae Nemoto, and William J Munro. "Aggregating quantum networks". Physical Review A **102**, 052613 (2020).

[33] A. B. U'Ren, K. Banaszek, and I. A. Walmsley. "Photon engineering for quantum information processing". Quantum Info. Comput. **3**, 480–502 (2003).

[34] Jürgen Brendel, Nicolas Gisin, Wolfgang Tittel, and Hugo Zbinden. "Pulsed energy-time entangled twin-photon source for quantum communication". Physical Review Letters **82**, 2594 (1999).

[35] I. Marcikic, H. de Riedmatten, W. Tittel, V. Scarani, H. Zbinden, and N. Gisin. "Time-bin entangled qubits for quantum communication created by femtosecond pulses". Phys. Rev. A **66**, 062308 (2002).

[36] Robert Thomas Thew, Sébastien Tanzilli, Wolfgang Tittel, Hugo Zbinden, and Nicolas Gisin. "Experimental investigation of the robustness of partially entangled qubits over 11 km". Physical Review A **66**, 062304 (2002).

[37] Pieter Kok, William J Munro, Kae Nemoto, Timothy C Ralph, Jonathan P Dowling, and Gerard J Milburn. "Linear optical quantum computing with photonic qubits". Reviews of Modern Physics **79**, 135 (2007).

[38] Alison M Yao and Miles J Padgett. "Orbital angular momentum: origins, behavior and applications". Advances in optics and photonics **3**, 161–204 (2011).

[39] YH Shih and AV Sergienko. "Observation of quantum beating in a simple beam-splitting experiment: Two-particle entanglement in spin and space-time". Physical Review A **50**, 2564 (1994).

[40] S Ramelow, L Ratschbacher, A Fedrizzi, NK Langford, and A Zeilinger. "Discrete tunable color entanglement". Physical review letters **103**, 253601 (2009).

[41] Sergei Slussarenko and Geoff J Pryde. "Photonic quantum information processing: A concise review". Applied Physics Reviews**6** (2019).

[42] Atharv Joshi, Kyungjoo Noh, and Yvonne Y Gao. "Quantum information processing with bosonic qubits in circuit qed". Quantum Science and Technology **6**, 033001 (2021).

[43] Yue Wu, Shimon Kolkowitz, Shruti Puri, and Jeff D Thompson. "Erasure conversion for fault-tolerant quantum computing in alkaline earth rydberg atom arrays". Nature communications **13**, 4657 (2022).

[44] Aleksander Kubica, Arbel Haim, Yotam Vaknin, Harry Levine, Fernando Brandão, and Alex Retzker. "Erasure qubits: Overcoming the t 1 limit in superconducting circuits". Physical Review X **13**, 041022 (2023).

[45] Mingyu Kang, Wesley C Campbell, and Kenneth R Brown. "Quantum error correction with metastable states of trapped ions using erasure conversion". PRX Quantum **4**, 020358 (2023).

[46] Takahiro Tsunoda, James D Teoh, William D Kalfus, Stijn J de Graaf, Benjamin J Chapman, Jacob C Curtis, Neel Thakur, Steven M Girvin, and Robert J Schoelkopf. "Error-detectable bosonic entangling gates with a noisy ancilla". PRX Quantum **4**, 020354 (2023).

[47] Chao-Yang Lu, Wei-Bo Gao, Jin Zhang, Xiao-Qi Zhou, Tao Yang, and Jian-Wei Pan. "Experimental quantum coding against qubit loss error". Proceedings of the National Academy of Sciences **105**, 11050–11054 (2008).

[48] Shuo Ma, Genyue Liu, Pai Peng, Bichen Zhang, Sven Jandura, Jahan Claes, Alex P Burgers, Guido Pupillo, Shruti Puri, and Jeff D Thompson. "High-fidelity gates and mid-circuit erasure conversion in an atomic qubit". Nature **622**, 279–284 (2023).

[49] Pascal Scholl, Adam L Shaw, Richard Bing-Shiun Tsai, Ran Finkelstein, Joonhee Choi, and Manuel Endres. "Erasure conversion in a high-fidelity rydberg quantum simulator". Nature **622**, 273–278 (2023).

[50] Harry Levine, Arbel Haim, Jimmy SC Hung, Nasser Alidoust, Mahmoud Kalaee, Laura DeLorenzo, E Alex Wollack, Patricio Arrangoiz Arriola, Amirhossein Khalajhedayati, Yotam Vaknin, et al. "Demonstrating a long-coherence dual-rail erasure qubit using tunable transmons" (2023).

[51] Kevin S Chou, Tali Shemma, Heather McCarrick, Tzu-Chiao Chien, James D Teoh, Patrick Winkel, Amos Anderson, Jonathan Chen, Jacob Curtis, Stijn J de Graaf, et al. "Demonstrating a superconducting dual-rail cavity qubit with erasure-detected logical measurements" (2023).

[52] G Alber, Th Beth, Ch Charnes, A Delgado, M Grassl, and M Mussinger. "Stabilizing distinguishable qubits against spontaneous decay by detected-jump correcting quantum codes". Physical Review Letters **86**, 4402 (2001).

[53] Thomas M Stace, Sean D Barrett, and Andrew C Doherty. "Thresholds for topological codes in the presence of loss". Physical review letters **102**, 200501 (2009).

[54] Sean D Barrett and Thomas M Stace. "Fault tolerant quantum computation with very high threshold for loss errors". Physical review letters **105**, 200502 (2010).

[55] Nicolas Delfosse and Gilles Zémor. "Linear-time maximum likelihood decoding of surface codes over the quantum erasure channel". Physical Review Research **2**, 033042 (2020).

[56] Alan Agresti and Brent A Coull. "Approximate is better than "exact" for interval estimation of binomial proportions". The American Statistician **52**, 119–126 (1998).

[57] Shin Nishio. "C++ implementation of multiplexed toric codes simulator". GitHub (2024). url: `https://github.com/parton-quark/Multiplexed_Toric`.

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

[59] Jean-Pierre Tillich and Gilles Zémor. "Quantum ldpc codes with positive rate and minimum distance proportional to the square root of the blocklength". IEEE Transactions on Information Theory **60**, 1193–1202 (2013).

[60] Nicholas Connolly, Vivien Londe, Anthony Leverrier, and Nicolas Delfosse. "Fast erasure decoder for hypergraph product codes". Quantum **8**, 1450 (2024).

[61] Nicholas Connolly and Shin Nishio. "Python implementation of multiplexed HGP codes simulator". GitHub (2024). url: `https://github.com/parton-quark/Multiplexed_HGP`.

[62] Mikael Afzelius, Christoph Simon, Hugues De Riedmatten, and Nicolas Gisin. "Multimode quantum memory based on atomic

frequency combs". Physical Review A **79**, 052329 (2009).

[63] Stephen B Wicker and Vijay K Bhargava. "Reed-solomon codes and their applications". John Wiley & Sons. (1999).

[64] Masahiro Hara, Motoaki Watabe, Tadao Nojiri, Takayuki Nagaya, and Yuji Uchiyama. "Two-dimensional code" (Japan Patent, 07-254037,A(1995)). Toyota Central Research & Development Lab Inc.

[65] JG Proakis and M Salehi. "Digital communications, vol. 1221" (1987).

[66] Shiro Kawabata. "Quantum interleaver: quantum error correction for burst error". Journal of the Physical Society of Japan **69**, 3540–3543 (2000).

[67] Michael G Luby, Michael Mitzenmacher, Mohammad Amin Shokrollahi, and Daniel A Spielman. "Efficient erasure correcting codes". IEEE Transactions on Information Theory **47**, 569–584 (2001).

A Encoding 2^2 dimensional quantum information in a photon

Here we give the transition of states in the circuit shown in Fig. 1 (B) and (C).

A.1 State transition for Fig. 1 (B)

$$
\begin{aligned}
& (\alpha|HS\rangle + \beta|HL\rangle + \gamma|VS\rangle + \delta|VL\rangle) \otimes |+\rangle \otimes |+\rangle \\
\xrightarrow{CX_{\mathrm{m_1,p}}} & \left(\frac{1}{\sqrt{2}}(\alpha|HS\rangle + \beta|HL\rangle + \gamma|VS\rangle + \delta|VL\rangle)|0\rangle + (\alpha|VS\rangle + \beta|VL\rangle + \gamma|HS\rangle + \delta|HL\rangle)|1\rangle)\right) \\
& \otimes |+\rangle \cdots ① \\
\xrightarrow{\mathrm{SW}} & \left(\frac{1}{\sqrt{2}}(\alpha|HS\rangle + \beta|VS\rangle + \gamma|HL\rangle + \delta|VL\rangle)|0\rangle + (\alpha|HL\rangle + \beta|VL\rangle + \gamma|HS\rangle + \delta|VS\rangle)|1\rangle)\right) \\
& \otimes |+\rangle \cdots ② \\
\xrightarrow{CX_{\mathrm{m_2,p}}} & \frac{1}{\sqrt{2}}((\alpha|HS\rangle + \beta|VS\rangle + \gamma|HL\rangle + \delta|VL\rangle)|00\rangle + (\alpha|VS\rangle + \beta|HS\rangle + \gamma|VL\rangle + \delta|HL\rangle)|01\rangle \\
& +(\alpha|HL\rangle + \beta|VL\rangle + \gamma|HS\rangle + \delta|VS\rangle)|10\rangle + (\alpha|VL\rangle + \beta|HL\rangle + \gamma|VS\rangle + \delta|HS\rangle)|1\rangle)|11\rangle) \\
& \cdots ③
\end{aligned}
$$

To obtain the state $\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$ one can apply the $II/IX/XI/XX$ gates on the matter qubits upon a corresponding photon measurement outcome $HS/VS/HL/VL$, respectively.

A.2 State transition for Fig. 1 (C)

We use the $\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$ as the initial state of the matter qubit m_1, m_2. The state transition is:

$$
\begin{aligned}
& |H\rangle \otimes (\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle) \\
\xrightarrow{CX_{\mathrm{m_2,p}}} & \alpha|H00\rangle + \beta|V01\rangle \\
& +\gamma|H10\rangle + \delta|V11\rangle \cdots ① \\
\xrightarrow{\mathrm{SW}} & \alpha|HS00\rangle + \beta|HL01\rangle \\
& +\gamma|HS10\rangle + \delta|HL11\rangle \cdots ② \\
\xrightarrow{CX_{\mathrm{m_1,p}}} & \alpha|HS00\rangle + \beta|HL01\rangle \\
& +\gamma|VS10\rangle + \delta|VL11\rangle \cdots ③
\end{aligned}
$$

Then one can measure matter qubits in X-basis to obtain the desired state.

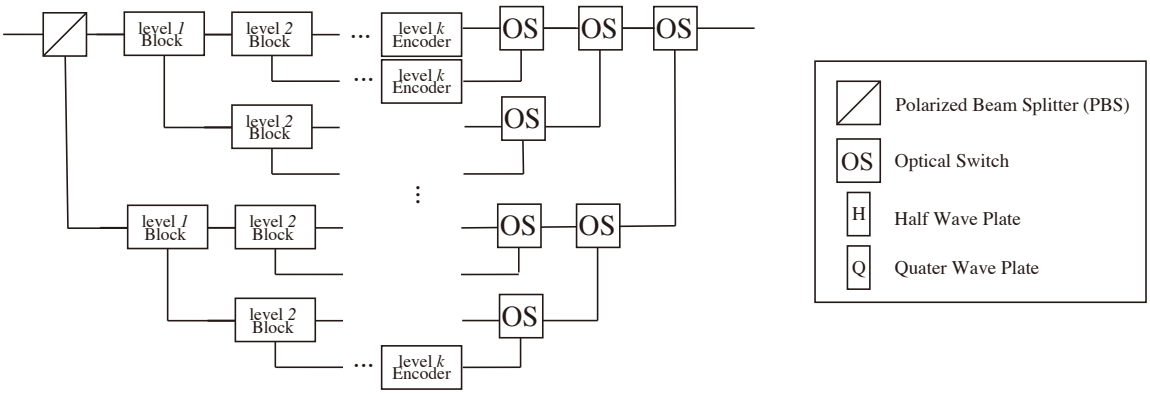


Figure 14: Optical circuit for preparing a photon with 2^k dimensional time-bin information.

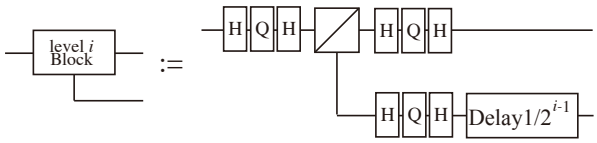


Figure 15: Level i block used in Fig. 14. It splits a single mode into two modes with a new component of time-bin DOF introduced to the state.

B Encoding 2^k dimensional quantum information in a photon

Here we show the circuit for encoding 2^k dimensional quantum information into time-bin in one photon in Fig. 14, where the level i block is defined as in Fig. 15. In the first half of the circuit, a level i block is used to introduce a new component of the time-bin by applying a delay. Then, in the middle of the circuit, each state corresponds to a different mode. In the second half of the circuit, multiplexed photons are output to one mode by applying the optical switches, forming a complete binary tree. This encoding circuit can prepare a photon with 2^k dimensional time-bin and 2 dimensional polarization in linear time with respect to k.

C Assignment strategies for surface codes

In this appendix, we describe the details and discussions on each assignment strategy for the surface code.

strategy i and ii: pair with minimum and maximum distance The Manhattan distance between qubits within the same photon is crucial when considering the impact of correlated errors. These strategies are deterministic and can be realized with simple calculations.

strategy iii: random Errors with strong correlation are similar to burst errors in classical communication in the sense that the errors have spatial locality. This locality of errors can be addressed by classical error-correcting codes using two methods. The first method treats multiple bits as a single symbol (an element of a finite field), such as BCH codes and Reed-Solomon codes [63]. Thanks to the high correctability of burst errors, Reed-Solomon codes are used in many classical systems, including QR codes [64], CDs, and satellite communications. Another method is the interleaving [65] technique. Interleaving eliminates locality by permuting the rows and columns of the code's generator matrix. There is also a method to apply interleaving to QECCs [66]. Inspired by interleaving, we have constructed two strategies for quantum multiplexing with randomness.

strategy iv: random + threshold The fourth strategy is a modified version of the third strategy. The pseudo-code is shown below in Algorithm 1. The flow of the algorithm is as follows: A "threshold" T is set as $2/d - 1$, which is the maximum distance between two qubits in the $[\![2d^2, 2, d]\!]$ surface code. This value will be used to check that the distance between the qubits in the same photon is high enough. Then, it randomly assigns qubits for each photon while respecting the distance threshold. It randomly selects the first qubit of the photon, then it randomly selects a qubit again and takes it as a candidate to assign it to this photon. When the distance between the candidate qubit and the qubit(s) already in the

photon is greater than the threshold, the qubit is accepted, and when it is less, it is rejected. This procedure is repeated until the photon has been assigned the appropriate number of qubits. If no qubit satisfies the threshold, then the threshold value is lowered by one. This ensures that the algorithm will always terminate. By repeating this process, we can assign all the qubits to photons. This strategy is designed to leverage randomness while also increasing the distance between qubits in the same photon.

This algorithm requires calculating the distance between two qubits, which is easy for the surface code thanks to the fact that the distance between qubits in the grid representation is defined by the taxicab metric (the Manhattan distance). Note that this and other assignment strategies can still be applied even if the number m of qubits per photon is not a divisor of the total number of qubits. In this case, we allow for a final "remainder" photon containing fewer than m qubits.

Strategy v: Stabilzier Error correction on the surface code is always considered up to multiplication by a stabilizer. This suggests that it may be useful to define photons using the qubit-support of a stabilizer check. Since the stabilizer generators for the surface code correspond to squares and crosses in the lattice, they have weight 4. On a $d \times d$ lattice, if d is divisible by 4, it will always be possible to partition the lattice into squares and crosses. In this perspective, the L-shapes used in the minimum-distance strategy can be thought of as "half-stabilizers" in the lattice. Since the usual strategy for converting an erasure problem into an error correction problem involves assigning erased qubits Pauli errors randomly, this stabilizer assignment strategy uses a mix of Z and X-type stabilizer generators from both squares and crosses. In this case, qubits are equally partitioned into the two types of stabilizers by tiling the lattice with alternating diagonal lines of squares and crosses.

Algorithm 1: Strategy iv. random + threshold

Input: $P = \{p_i\}$ (the set of photons) where initially $p_i = \{\emptyset\}$ (the set of qubits to be encoded in the ith photon), $Q = \{q_j\}$ (the list of all physical qubits in the code), and the multiplexing number m of qubits in each photon.

Output: $P = \{p_i\}$ (a set of sets of qubits assigned to the ith photon).

```
Initialize the threshold with T := d/2 − 1;
for photon p_i ∈ P do
    Pick a qubit q_j ∈ Q randomly;
    Move q_j from Q to p_i;
    while |p_i| < m do
        while |p_i| < m and Q ≠ ∅ do
            Pick a candidate qubit q_k ∈ Q at random;
            if the distance between q_k and all qubits in p_i is greater than T then
                Move q_k from Q to p_i;
            else
                Move q_k from Q to a waiting list Q';
        Move all qubits in Q' to Q;
        Update T := T − 1;
Return P;
```

D Peeling Decoder

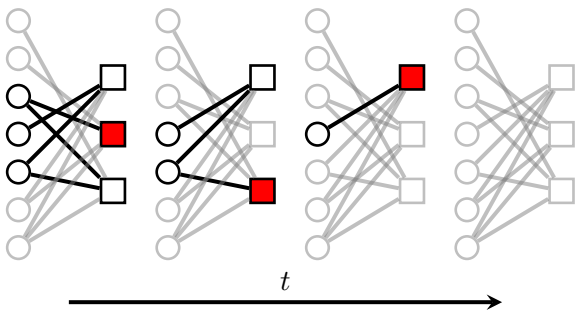
The *peeling decoder* refers to a linear-complexity erasure decoding algorithm originally designed for classical codes [67]. This algorithm corrects an erasure error by examining the subgraph of the Tanner graph corresponding to erased bits, whereby degree-1 check nodes in this subgraph give perfect information about adjacent bit nodes. Although not a maximum-likelihood decoder, the peeling decoder works well for codes with sparse Tanner graphs, such as LDPC codes. Because this algorithm only uses the Tanner graph, it can be directly applied to CSS codes as well. In this section, we briefly summarize the peeling decoder algorithm and several of its variations, beginning with a review of the classical erasure setting.

D.1 Peeling Algorithm for Classical Codes

For a classical code, an erasure error on a codeword can be modeled as the loss of a known subset of bits. By assigning these erased bits the values 0 or 1 at random and then making a syndrome measurement, the erasure correction problem can be converted into an error correction problem. Un-

like standard error correction, we make the additional assumption that non-erased bits do not have errors. Hence, it is sufficient to consider error correction using the subcode corresponding only to erased bits. In terms of the Tanner graph, this is equivalent to considering the subgraph induced by the erasure (consisting of the subset of erased bit-nodes and any adjacent check-nodes). The peeling algorithm is defined in terms of this erasure-induced subgraph.

A check is said to be *dangling* if it has degree 1 in the subgraph (i.e. a dangling check is adjacent to exactly one erased bit). Recall that each check-node corresponds to a position in the syndrome vector for the randomly selected erasure-supported error. The value of the syndrome bit corresponding to a dangling check gives perfect information about the error on the adjacent erased bit. Based on this, the error on this bit can be corrected and then removed from the original set of erased bits, thus shrinking the erasure-induced subgraph and possibly introducing new dangling checks. The peeling decoder functions by performing a sequence of partial corrections, one erased bit at a time, hence "peeling" the subgraph until no dangling checks remain. This process is shown visually in Fig. 16 and briefly summarized in Algorithm 2.



$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Figure 16: Example of the peeling algorithm applied to an erasure-induced subgraph of the Tanner graph of the classical code $C = \mathrm{Ker}(H)$, where circle-nodes denote bits and square-nodes denote checks. Gray edges and nodes are not included in the erasure. Red squares indicate a dangling check (degree 1 check-node in the subgraph). At each time step, a dangling check and adjacent erased bit are removed from the erasure, until the erasure is empty.

The decoding is successful if all erased bits have been corrected. A decoding failure occurs when all dangling checks have been peeled, but the remaining erasure is nonempty (i.e. all remaining checks in the subgraph have degree 2 or higher). Such a configuration is referred to as a *stopping set*. If an erasure pattern contains a stopping set, then the peeling algorithm will fail to find a correction. In particular, because the algorithm may not even return to the code space, this shows that peeling is not a maximum-likelihood decoder.

Algorithm 2: Peeling Algorithm

Input: A code $\mathrm{Ker}(H)$ with Tanner graph G, a set of erased bits E, and a syndrome vector s.

Output: A predicted error $\hat{e} \subseteq E$ such that $H\hat{e} = s$, or **Failure**.

```
Initialize ê = ∅;
while E ≠ ∅ do
    Compute erasure subgraph G_E ⊆ G;
    if ∃ dangling check s_i ∈ G_E then
        if s_i is unsatisfied then
            ∃ error on adjacent bit b_j ∈ E;
            Flip bit b_j, update syndrome s;
            Update ê := ê ∪ {b_j};
        else
            No error on adjacent bit b_j;
        Update E := E \ {b_j};
    else
        Return Failure;
Return ê;
```

D.2 Peeling Algorithm for CSS Codes

Erasure correction for a quantum code is modeled similarly to the classical case, with an erasure error on a codeword corresponding to the loss of a known subset of qubits. As in the classical case, erasure correction can be converted into error correction, with the modified rule that erased qubits are assigned Pauli errors in $\{I, X, Z, Y\}$ at random in the quantum case. For a CSS code, errors can be corrected by applying the peeling algorithm two times, once using the classical Tanner graph for H_Z and once again for H_X. Since X- and Z-type Pauli errors are corrected independently, the same initial erasure pattern is used both times.

D.3 Peeling Algorithm for Surface Codes

The *surface code peeling decoder* refers to a generalization of this algorithm adapted to surface codes [55], which uses additional information about stabilizers in the code. Before applying the standard peeling algorithm, the modified algorithm first computes a certain acyclic subgraph of the usual erasure-induced subgraph. By leveraging stabilizer equivalences, it is sufficient to apply the peeling algorithm only to this acyclic subgraph to correct the entire erasure error; the random values assigned to erased qubits not included in this subgraph are assumed to be correct. The advantage here is that an acyclic graph does not contain stopping sets; the peeling algorithm will always successfully terminate with a predicted erasure correction when applied to the acyclic subgraph in question. Hence, unlike the standard peeling decoder, the surface code peeling decoder is maximum-likelihood.

It remains to comment on how we compute this acyclic subgraph of the erasure-induced subgraph of the Tanner graph for the surface code. To explain this, we consider the usual depiction of a distance d surface code on a $d \times d$ lattice, whereby qubits are identified with edges in the lattice and X- and Z-type stabilizer checks are identified with vertices and plaquettes, respectively. That is say, the H_X-computed syndrome for Z-type Pauli errors on qubits is visualized by the subset of vertices corresponding to unsatisfied X-checks. A similar visualization for X-type Pauli errors is possible using the dual graph of this lattice picture. In the context of an erasure error, a subset of erased qubits is visualized by a corresponding set of erased edges in the surface code lattice. This erasure can also be thought of as the subgraph of the lattice consisting of erased qubit-edges and any vertices adjacent to these edges (not to be confused with the related erasure-induced subgraph of the Tanner graph).

After assigning erased qubits Pauli errors at random, as usual, we consider the correction of Z- and X-type errors independently. In the Z-error case, the syndrome corresponding to the unsatisfied X-checks is a subset of the vertices in the erasure-induced subgraph of the lattice. The algorithm proceeds by computing a spanning tree of the erasure-induced subgraph of the lattice (or a spanning forest in the case of a disjoint subgraph). This spanning tree in the lattice also corresponds to a subgraph in the Tanner graph of H_X; each leaf in the spanning tree corresponds to a dangling check in the subgraph. In this way, we obtain the acyclic subgraph of the erasure-induced subgraph of the Tanner graph mentioned earlier. Any two spanning trees are equivalent up to multiplication by stabilizers, as are the predicted errors obtained via the peeling algorithm. The X-errors are corrected in exactly the same way, except using the dual lattice.

This modified peeling algorithm is a linear-complexity, maximum-likelihood decoder for the surface code. We use this algorithm in our numerical simulations for the surface code. Our implementation of the surface code peeling decoder is available in [57]. This process is briefly summarized in Fig. 17 and 18.

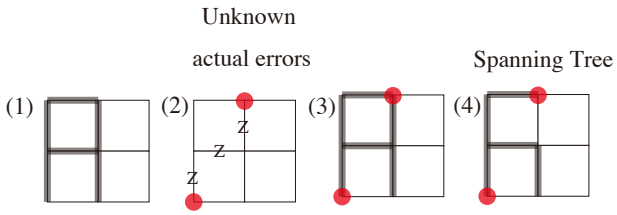


Figure 17: Illustration of the error correction process for an example of erasure errors with the surface code lattice. (1) Erased qubits are shown in bold grey lines. (2) Erasure errors are converted to random Pauli errors by replacing erased qubits with mixed states. The syndrome (indicated by the red vertices) is then computed by applying stabilizer measurements as explained in Sec. 2.2. (3) Information seen by the decoding algorithm: erasure pattern and syndrome. (4) A spanning tree for the erasure pattern in the lattice is computed; this is identified with a corresponding acyclic subgraph of the Tanner graph. The *surface code peeling decoder* then corrects the qubits one by one using this subgraph.

D.4 Peeling Algorithm for HGP Codes

The *pruned peeling + VH decoder* [60] is yet another generalization of the peeling decoder adapted to the special case of HGP codes. As mentioned in Appendix D.2, because these are a type of CSS code, the standard peeling algorithm can be directly applied to HGP codes. However, this algorithm performs very poorly in practice, even for LDPC codes. This poor performance can be explained by the presence of stopping sets unique to HGP codes which have no analogue in the classical case. These stopping sets can be grouped into two types: *stabilizer* and *classical*,

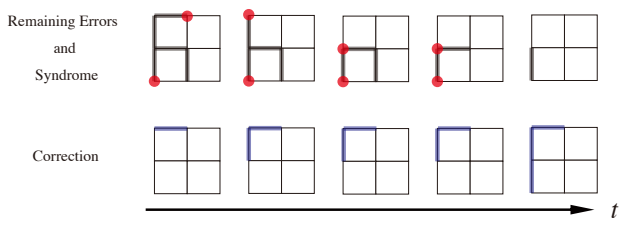


Figure 18: Peeling process to decode the erasure error pattern given in Fig. 17. Grey edges indicate the spanning tree in the surface code lattice and red vertices indicate the syndrome. The peeling algorithm is applied to the corresponding subgraph of the Tanner graph. Blue edges indicate corrections applied to qubits. Each time step denotes one iteration of the peeling algorithm, whereby the erasure is reduced by one qubit.

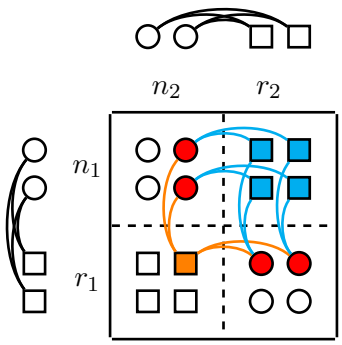


Figure 19: An example of a stabilizer stopping set for H_z of the 2×2 surface code. The red circles indicate an erasure error, which consists of all of the qubits adjacent to the X-stabilizer denoted by the orange square. The cyan squares denote the Z-stabilizers adjacent to this set of erased qubits. Notice that each of these Z-stabilizers is adjacent to an even number of erased qubits, which is a consequence of the commutation relation between X- and Z-type stabilizers and also guarantees that there are no peelable degree-1 checks in the erasure subgraph.

both of which cause the decoder to fail. The pruned peeling + VH decoder is designed to address these stopping sets.

Stabilizer stopping sets occur when the erasure contains the qubit-support of an X- or Z-type stabilizer. Recall that the X- and Z-type stabilizers must commute, which is possible if and only if they overlap on an even number of qubits. Hence, in terms of the nodes denoting stabilizer generators in the Tanner graph, if we fix a node corresponding to an X-stabilizer generator and consider the set of qubits on which it is supported, each Z-stabilizer generator with overlapping qubit-support must overlap on an even number of qubits in order to satisfy the commutativity condition. Since the nodes denoting Z stabilizer generators are exactly the check nodes in the Tanner graph of H_Z, these constitute a stopping set, as shown in the example of Fig. 19. In particular, there exist no dangling checks (which have degree 1) and hence this is a peeling decoder-stopping set. A similar relationship holds for Z-stabilizers in the Tanner graph of H_X.

Such a stopping set can be modified by fixing a value at random for one qubit of the stabilizer and removing this qubit from the erasure. This reduces the degree of a single check-node in the erasure subgraph by 1, possibly introducing a dangling check and allowing the standard peeling algorithm to become unstuck. Removing a qubit from the erasure is equivalent to declaring the random mixed state on this qubit to be correct. This technique is valid for CSS codes because there exists a solution on the remaining erased qubits in the stabilizer-support such that the combined contribution to the error is at most a stabilizer. This procedure, known as *pruned peeling*, is applicable to any CSS code, not just HGP codes.

Classical stopping sets are another common type of peeling decoder stopping set which are only defined for HGP codes. These refer to patterns of erased qubits supported entirely on a single row or column in the HGP Tanner graph block structure of Fig. 10. In the simplest case, a peeling decoder stopping set for one of the classical codes used in the HGP construction lifts to a classical stopping set for the HGP code. Furthermore, any HGP peeling decoder stopping set can be decomposed into a union of vertical and horizontal sets on the columns and rows of the Tanner graph; although we refer to these components as *classical stopping sets*, a single component in isolation need not be a stopping set for the corresponding classical code.

The *VH decoder* algorithm functions by ordering and efficiently solving each of these classical stopping sets in sequence, when possible, using the Gaussian decoder. The basic premise relies on the fact that, for a HGP code of length N, the component classical codes have length on the order of $\sqrt{N}$. Hence, even though Gaussian elimination (which has cubic complexity in the code length) is usually too slow for practical use, the complexity is reduced when restricted to a single classical stopping set. However, classical stopping sets often overlap (i.e. share a check-node in

the Tanner graph), in which case the two stopping sets cannot be resolved independently without introducing some additional restrictions. By checking these conditions, the VH decoder attempts to find solutions for classical stopping sets which are compatible in these overlapping cases. If such a solution is found for each classical stopping set, these combine to give a solution for the HGP code. However, there exist erasure configurations where the VH decoder becomes stuck as well. In general, these will occur when there exist cycles of classical stopping sets in the erasure-induced subgraph.

The *pruned peeling + VH decoder* refers to the combination of these three strategies (standard peeling, correction of stabilizer stopping sets, and correction of classical stopping sets). For simplicity, we also use the term *combined decoder* to refer to peeling + pruned peeling + VH. A stopping set for the combined decoder meets three conditions: there exist no remaining dangling checks; the remaining erasure does not cover the qubit-support of a stabilizer; remaining classical stopping sets form a cycle. Although these happen infrequently at a low erasure rate, an erasure pattern of this form will result in a decoder failure. These are distinct from logical errors, which can only be identified in numerical simulations where the decoding algorithm successfully terminates. The maximum-likelihood decoder always terminates, and thus, logical errors are the only source of failures. Although we make a distinction between these two possibilities, we will use the term *error recovery failure* to refer to either a decoder failure or a non-decoder-failure logical error. A more detailed discussion of these differences is included in Appendix E.2.

The computational complexity of the combined decoder is dominated by the step applying cubic-complexity Gaussian elimination to classical stopping sets (peeling and pruned peeling both have linear complexity). The number of classical stopping sets is on the order of the number of rows and columns in the Tanner graph ($\sqrt{N}$ for an HGP code of length N). Furthermore, since classical stopping sets have a size of approximately $\sqrt{N}$, the effect of Gaussian elimination on a single classical stopping set contributes $O(N^{1.5})$ to the complexity. This becomes $O(N^2)$ across all classical stopping sets, establishing this algorithm as a quadratic complexity decoder for HGP codes.

E Additional Details for HGP Codes

E.1 Symmetric Constructions

Recall equations 3 and 4 used to define the parity check matrices for HGP codes. The the sizes of the matrices H_X and H_Z obtained in this way are determined by the sizes of the input classical matrices H_1 and H_2. Hence, the number of qubits and stabilizer checks are controlled by the size of the input classical matrices; equations 5 and 6 give the exact dimensions for H_X and H_Z obtained from matrices $H_1 = [r_1 \times n_1]$ and $H_2 = [r_2 \times n_2]$. Referring to the Tanner graph of Fig. 10, the $n_1 \times n_2$ and $r_1 \times r_2$ blocks denote the qubit nodes and the $r_1 \times n_2$ and $n_1 \times r_2$ blocks denote the X- and Z-type stabilizer generators, respectively.

Choosing classical matrices of the same size ensures an equal number of stabilizer checks in the HGP code, but a biased code can also be constructed by using matrices of different sizes. Furthermore, using $H_2 = H_1^T$ yields a symmetric construction for H_X and H_Z and guarantees that the two blocks of qubits in this product graph picture are squares of equal size. In our numerical simulations, we consider two types of HGP code construction: an *equal block* case coming from the symmetric construction with $H_2 = H_1^T$, whereby $r_2 = n_1$ and $n_2 = r_1$; and a *non-equal block* case using different matrices H_1 and H_2 of the same size, so $r_1 = r_2 = r$ and $n_1 = n_2 = n = 2r$ (we make a choice to use matrices with half as many rows as columns).

E.2 Types of Error Recovery Failures for the Pruned Peeling + VH Decoder

In Sec. 5.2 and Sec. D.4, we observed that the combined decoder (peeling + pruned peeling + VH) is not maximum-likelihood since decoder failures are possible in addtion to logical errors. Failure rate in the literature usually refers to the logical error rate, which is the only source of errors for a maximum-likelihood decoder. Logical errors for the erasure channel can only occur when the erasure covers a logical code word. However, there may exist erasure patterns covering a logical error which result in a decoder failure, and hence are not properly identified as logical errors. This distinction is stated visually by the Venn diagram of Fig. 20. The failure rate computed in our nu-

merical simulations for the *combined decoder* is the cummulative effect of these two possibilities, what we refer to as *error recovery failure rate* on the vertical axis in the plots of our numerical simulations for HGP codes. Note that failures at low erasure rates are almost exclusively due to logical errors, and so this distinction can be regarded as negligible in the practical regime.

Note that peeling + pruned peeling is theoretically a maximum likelihood decoder in the special case of the surface code. This is equivalent to the spanning-tree-based ML decoder for the surface code [55]. However, our implementation of pruned peeling is not perfect since it cannot identify the support of an arbitrary erased stabilizer. For the *combined decoder*, the simplest classical stopping sets correspond to a fully erased row or column in the HGP Tanner graph. These are exactly the stopping sets of a repetition code, coinciding with logical errors for the surface code. In general, there do not exist erasure patterns giving a VH decoder failure for the surface code which do not also cover a logical error.

Figure 11 shows the performance of the *combined decoder* applied to the 10×10 surface code. Comparing this to Fig. 4, which uses the ML decoder for the same surface code, we see a noticeable degradation in performance. This gap is explained by the existence of decoder failures in the combined case which do not exist for the ML decoder. Furthermore, the failure rate of the combined decoder converges to 1 as the erasure rate goes to 1, in contrast with the convergence to 0.75 for the ML decoder. This is because the erasure pattern is always a VH decoder stopping set when all qubits are erased, guaranteeing a decoder failure. Since there are no stopping sets in the ML case, however, a 100% erasure rate is equivalent to generating a uniformly random physical Pauli error on the code. We see a convergence to 0.75 logical error rate because this error is identity 25% of the time.

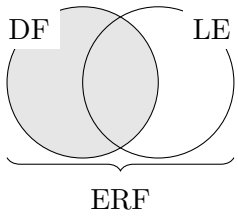


Figure 20: Venn diagram distinguishing between the types of failures possible using the pruned peeling + VH decoder. A *decoding failure* (DF) occurs when the decoder becomes stuck in a stopping set it cannot correct, in which case the algorithm fails to terminate. A *logical error* (LE) occurs when the set of erased qubits covers a logical operator. An *error recovery failure* (ERF) refers to the union of these two possibilities. It is possible that an application of the decoding algorithm to an erasure pattern covering a logical operator successfully returns to the code space, but results in a logical error; this possibility is represented in the Venn diagram by LE\DF. Alternatively, DF∩LE denotes those stopping sets for the decoder that cover a logical operator while DF\LE denotes those stopping sets that do not. In all three cases, the decoder fails to recover the error.

E.3 Technical Details for Assignment Strategies used with Multiplexed HGP Codes

Strategy ii. stabilizer The stabilizer strategy was initially introduced in Sec. 4 for the surface code, but it can be applied to CSS codes more generally. The erased qubit-support of a stabilizer will be a peeling decoder stopping set, but these are precisely the stopping sets that the pruned peeling algorithm attempts to correct. Hence, this strategy is motivated by the idea that losing a photon corresponding to a single stabilizer individually induces a correctable erasure pattern.

In the stabilizer strategy for HGP codes, we partition the qubits into sets corresponding to the qubit-support for disjoint stabilizers. Qubits are assigned to stabilizers based on these sets. The number of qubits per stabilizer is fixed for an LDPC code and matches the row weight of H_X or H_Z. Depending on the multiplexing number, the photons can also represent partial stabilizers or multiple stabilizers.

In the special case of the surface code with a $d \times d$ lattice, where d is divisible by 4, it is always possible to partition the qubits into a combination of disjoint X- and Z-type stabilizer generators as seen in Fig. 6, each of which is supported on 4 qubits. For a more general HGP code, we may attempt a similar assignment strategy by identifying the qubit-support of the stabilizer generators from the rows of H_X and H_Z. However, we cannot guarantee that a partition of qubits into disjoint stabilizers is possible without placing constraints on the number of qubits and the row and column weights in the parity check matrices. Instead, we adopt an imperfect but simpler strategy for generic HGP codes,

which does not require any additional assumptions about the code except that H_X and H_Z are LDPC. This strategy can be used with stabilizers coming only from H_X, only from H_Z, or a combination of both, provided that these matrices have the same row weight. Note that restricting to a single type of stabilizer creates a bias in the error correction, as was commented in the surface code case.

The first step of this strategy is to search for a partition of the qubits into disjoint stabilizers. To do this, we begin by choosing a row at random from H_X or H_Z; the nonzero entries in this row represent the qubit-support of a single stabilizer. We then eliminate any overlapping stabilizers by deleting the rows from the matrices that share columns with nonzero entries with the previously selected row. Then we repeat this strategy until either all qubits have been divided into disjoint stabilizers or we exhaust the remaining rows that do not overlap with our previous selections. The result is that as many qubits as possible have been divided into non-overlapping sets corresponding to the qubit-support of disjoint stabilizers, possibly with some remaining ungrouped qubits.

Finally, the qubits are assigned to photons based on the disjoint sets identified in the previous step. Ordering the qubits by their stabilizer assignments, we then redistribute these into photons. The remaining ungrouped qubits are assigned after exhausting the chosen stabilizers. When the multiplexing number matches the stabilizer weight (that is, the row weight of H_X or H_Z), each photon ideally matches a stabilizer, possibly with some remainder photons at the end for the ungrouped qubits. When the multiplexing number matches a fraction or multiple of the stabilizer weight, then the photons represent a partial stabilizer or multiple stabilizers, respectively. Allowing for the leftover qubits at the end ensures that this strategy can be applied with various multiplexing numbers, even when a perfect partition of qubits into stabilizers is not found. Because stabilizers are selected at random, this assignment strategy can be understood as a combination of the random and stabilizer strategies introduced before.

Strategy iii. sudoku The VH decoder is designed to address classical stopping sets for the peeling decoder, but there exist combinations of classical stopping sets that cannot be solved using this technique and result in a decoder failure. However, we may reduce the likelihood of a decoder failure by reducing the number of classical stopping sets in general. Classical stopping sets are supported on a single row or column of the HGP code Tanner graph. Thus, we propose an assignment strategy based on choosing qubits in the same photon from different rows and columns. The method for doing this is outlined in Algorithm 3.

Algorithm 3: Strategy iii. sudoku

Input: $P = \{p_i\}$ (the set of photons, where p_i is the set of qubits in photon i), $Q = \{q_j = (r_j, c_j, b_j)\}$ (a list of 3-tuples with the row, column, and block of each physical qubit in the HGP code), and the number m of qubits per photon.

Output: $P = \{p_i\}$ (photon assignments).

```
for photon p_i ∈ P do
    Pick a qubit q_j ∈ Q at random;
    Move q_j from Q to p_i;
    while |p_i| < m and Q ≠ ∅ do
        Pick a candidate qubit q_k ∈ Q;
        if q_k is in a different row and column (or block) from each
          previously selected q_j ∈ p_i ((r_k ≠ r_j and c_k ≠ c_j) or b_k ≠ b_j)
          then
            Move q_k from Q to p_i;
        else
            Move q_k from Q to a temporary waiting list Q';
    Move all qubits in Q' back to Q;
    while |p_i| < m do
        Pick a qubit q_k ∈ Q at random;
        Move q_k from Q to p_i;
Return P;
```

This strategy assumes that the number of qubits per photon does not exceed the minimum length of a row or column in the Tanner graph, although this condition may be relaxed by instead allowing for a minimal number of qubits from the same row or column to be added to the same photon. Qubits are assigned to photons at random, checking that each newly added qubit is not supported on the same row or column as any qubit already assigned to a given photon. In the case of a fixed number of photons where no valid qubit

assignments remain, we drop the condition and default to random assignment.

Strategy iv: row-col Although not a practical assignment strategy, the case where only qubits from the same row or column of the HGP code Tanner graph are assigned to the same photon is of theoretical interest. This strategy attempts to maximize the number of classical stopping sets resulting from photon loss and thus increase the likelihood of a VH decoder failure. Verifying that this assignment strategy performs very poorly in numerical simulations serves as a proof of concept for the VH decoder and also justifies the preferred strategies using qubits from different rows and columns.

Fig. 21 shows the performance of this strategy for a ⟦512,8⟧ HGP code at several multiplexing numbers. Although surprisingly the $m = 2$ case seems to outperform the no-multiplexing case, the failure rate otherwise increases as m increases. Failures of the VH decoder are the result of certain configurations of classical stopping sets, and hence increasing the latter also increases the former. In particular, this explains the dramatic jump between the $m = 8$ and $m = 16$ cases. Since the blocks in this code's Tanner graph are 16×16, each photon in the $m = 16$ case corresponds to an entire row or column. Loss of any photon yields a classical stopping set, and hence VH decoder failures are common. This also confirms the significance of designing assignment strategies to avoid classical stopping sets in our simulations of HGP codes. In general, we expect the performance of the row-column strategy to drop significantly as m becomes equal to or larger than the side length of the block in the HGP Tanner graph.

Strategy v. diagonal Whereas the sudoku strategy assigns qubits at random subject to the condition of being in a different row or column, qubits in the HGP code Tanner graph may also be grouped diagonally within each block. In this way, it is possible to satisfy the sudoku condition without relying on randomness. A $d \times d$ grid can be divided into d non-overlapping diagonal slices, where we allow slices to wrap around. Since no two qubits in the same diagonal slice are contained in the same row or column of the grid, this technique also guarantees that we avoid classical stopping sets within a single photon. Photon assignment is thus based on grouping together the qubits in the same diagonal slice. Each of the two

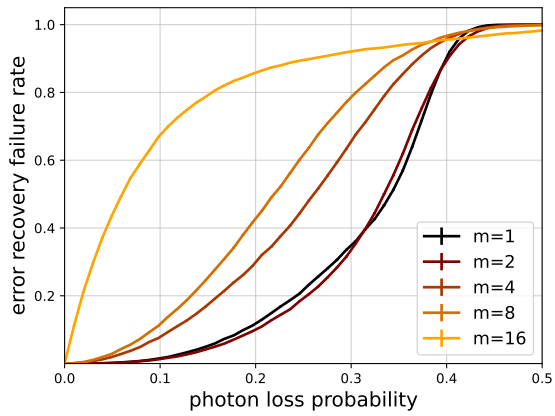


Figure 21: The performance for multiplexed communication with a ⟦512,8⟧ equal-block (16×16) HGP code obtained from the symmetric construction using $r = n = 16$ with assignment strategy (iv) row-column. Increasing the number of qubits per photon using this strategy rapidly increases the failure rate. At $m = 16$, each photon corresponds to an entire row or column in the HGP Tanner graph, whereby losing even one photon guarantees a classical stopping set.

qubit-squares in the HGP code Tanner graph is considered separately, but if we require that the ratio of the squares' side lengths is a whole number, then the qubits can be cleanly partitioned into photons of size matching the side length of the smaller square. HGP codes with rectangular Tanner graph block sizes can also use the diagonal strategy, provided that the length of the diagonal slice does not exceed the length of the shortest side. If longer slices are permitted in the rectangular case, then instead a minimal number of qubits in the same row or column are allowed.

The implementation of this strategy as described in Algorithm 4 is simple, provided one precomputes a *diagonal ordering* on the qubits in the HGP Tanner graph. Referring to the block structure of Fig. 10, the qubits in a given block are indexed along the non-overlapping diagonal slices. These slices are allowed to wrap around the sides of the square, which guarantees that no two qubits in the same slice are contained in the same row or column. The qubits in the second block are indexed sequentially after the first block. In our numerical implementation, a separate function to compute this ordering on the qubits in an HGP code is used along with the assignment function.

Algorithm 4: Strategy v. diagonal

Input: $P = \{p_i\}$ (the set of photons where p_i is the set of qubits in photon i), $Q = \{q_j\}$ (a list of physical qubits ordered along the diagonal), and the number m of qubits per photon.

Output: $P = \{p_i\}$ (photon assignments).

```
for photon p_i ∈ P do
    for qubits with indices j ∈ {im, ···, (i+1)m} do
        Move q_j from Q to p_i
return P;
```