Spanning Tree Matching Decoder for Quantum Surface Codes

Diego Forlivesi, *Graduate Student Member, IEEE*, Lorenzo Valentini, *Member, IEEE*, and Marco Chiani, *Fellow, IEEE*

***Abstract*— We introduce the spanning tree matching (STM) decoder for surface codes, which guarantees the error correction capability up to the code's designed distance by first employing an instance of the minimum spanning tree on a subset of ancilla qubits within the lattice. Then, a perfect matching graph is simply obtained, by selecting the edges more likely to be faulty. A comparative analysis reveals that the STM decoder, at the cost of a slight performance degradation, provides a substantial advantage in decoding time compared to the minimum weight perfect matching (MWPM) decoder. Finally, we propose an even more simplified and faster algorithm, the Rapid-Fire (RFire) decoder, designed for scenarios where decoding speed is a critical requirement.**

***Index Terms*— Quantum error correcting codes, quantum communications, quantum computing, surface codes.**

I. Introduction

LEVERAGING the unique characteristics of quantum mechanics has considerably expanded the possibilities within the realm of information management, encompassing various domains such as sensing, processing, and communication [1], [2], [3]. The key hurdle in constructing a quantum computer is the inevitable presence of errors that, if not addressed, rapidly degrade quantum information. Therefore, error correction is crucial for meaningful quantum computation [4], [5], [6], [7], [8]. Surface codes are considered central to the architecture for the first generation of quantum computers, thanks to their high error thresholds, planar structure, and locality [9], [10]. The minimum weight perfect matching (MWPM) decoder is presently the most widely used decoder for surface codes [11], [12], [13]. This decoder results in large threshold error rates, but in practice, its polynomial time complexity introduces a latency that can be a bottleneck for fault-tolerant quantum computing architectures [5], [14], [15]. Considerable efforts have been invested in optimizing the performance of this decoder [16], [17]. Sub-optimal algorithms based on the union-find (UF) decoder have also been proposed, achieving almost linear time in code length [18], [19]. Additionally, there is ongoing research into neural network-based solutions [20].

Manuscript received 21 February 2024; revised 8 April 2024; accepted 29 April 2024. Date of publication 6 May 2024; date of current version 12 July 2024. This work was supported in part by the European Union - Next Generation EU, under PNRR project PRIN n. 2022JES5S2. The associate editor coordinating the review of this letter and approving it for publication was W. Mesbah. *(Corresponding author: Marco Chiani.)*

The authors are with the Department of Electrical, Electronic, and Information Engineering "Guglielmo Marconi" and the CNIT/WiLab, University of Bologna, 40136 Bologna, Italy (e-mail: diego.forlivesi2@unibo.it; lorenzo.valentini13@unibo.it; marco.chiani@unibo.it).

Digital Object Identifier 10.1109/LCOMM.2024.3396882

In this letter, we present a fast decoding technique tailored for surface codes, called the spanning tree matching (STM) decoder. The STM algorithm involves implementing an instance of the minimum spanning tree (MST) on a subset of the ancilla qubits within the lattice. This is followed by a simple and fast construction of a perfect matching graph, resulting in the estimated error pattern. Finally, we propose an even more simplified and faster algorithm, the RFire decoder, tailored for situations where decoding speed is of paramount importance. We conduct a comparative analysis of the performance, considering logical error rates and execution times, between the STM, RFire, and MWPM decoders. We show that, at the price of some performance degradation, the proposed algorithms offer significant advantages in terms of decoding time.

II. Preliminaries and Background

A. Quantum Stabilizer Error-Correcting Codes

The Pauli operators are denoted as $\boldsymbol{X}, \boldsymbol{Y}$, and $\boldsymbol{Z}$. We indicate with $[[n, k, d]]$ a quantum error correcting code (QECC) encoding k logical qubits $|\varphi\rangle$ into a codeword of n data qubits $|\psi\rangle$, with minimum distance d. The code allows the correction of all patterns with up to $t = \lfloor (d-1)/2 \rfloor$ data qubit errors.

Employing the stabilizer formalism, each code is characterized by $n-k$ independent and commuting operators $\boldsymbol{G}_i \in \mathcal{G}_n$, termed stabilizer generators or simply generators, with $\mathcal{G}_n$ being the Pauli group on n qubits [5], [6]. The subgroup of $\mathcal{G}_n$ generated by all combinations of the $\boldsymbol{G}_i \in \mathcal{G}_n$ is called stabilizer and indicated as $\mathcal{S}$. The code $\mathcal{C}$ is the set of quantum states $|\psi\rangle$ stabilized by $\mathcal{S}$, i.e., satisfying $\boldsymbol{S}|\psi\rangle = |\psi\rangle \ \forall \boldsymbol{S} \in \mathcal{S}$, or, equivalently, $\boldsymbol{G}_i|\psi\rangle = |\psi\rangle, i = 1, 2, \ldots, n-k$. The operators that commute with the stabilizer group but are not part of it are called logical operators. The generators specify measurements on quantum codewords that do not disturb the original quantum state. These measurements are carried out using extra qubits, named ancillas. In fact, assume an error $\boldsymbol{E} \in \mathcal{G}_n$ affects a codeword, so that the state becomes $\boldsymbol{E}|\psi\rangle$. It is possible to extract a binary sequence $\boldsymbol{s}$ (also referred to as error syndrome) where the i-th entry s_i is zero if $\boldsymbol{G}_i$ commutes with $\boldsymbol{E}$, while $s_i = 1$ if $\boldsymbol{G}_i$ anticommutes with it. This enables the possibility to perform quantum error correction through error syndrome decoding using as input the binary sequence $\boldsymbol{s}$. We will refer to ancillas measuring $s_i = 1$ as *defects*.

Among stabilizer codes we find the surface codes. These codes arrange qubits on a planar sheet [9], [21], [22], [23]. One of the advantages of such an arrangement, is that it requires only nearest-neighbor interactions between qubits. Also, this

structure enables the possibility to perform a single round of stabilizer measurements with parallel operations [24]. Logical operators can be easily identified on the surface codes: $\boldsymbol{Z}_L$ ($\boldsymbol{X}_L$) operator consists of a tensor product of $\boldsymbol{Z}$'s ($\boldsymbol{X}$'s) crossing horizontally (vertically) the lattice [21]. In the case of a QECC over the depolarizing channel with data qubit error probability $p \ll 1$, we can approximate the logical error rate as [25]

$$p_{\mathrm{L}} \approx (1-\beta_{t+1}) \binom{n}{t+1} p^{t+1} \tag{1}$$

where β_{t+1} is the fraction of errors of weight $t+1$ that the decoder is able to correct.

B. Minimum Weight Perfect Matching

A key characteristic of surface codes is the availability of a minimum weight decoder, known as the MWPM. This syndrome decoder constructs a graph in which vertices represent defects, and edges are assigned weights depending on the error probability of the qubits separating the pair. For instance, in case of independent identically distributed (i.i.d.) data qubit errors, these weights correspond to the number of qubits between the pair. In general, such a weighting procedure is performed using the Dijkstra algorithm to connect the defects on the graph representing the full lattice [11]. By suitably assigning distances on the lattice, Dijkstra's algorithm allows for the consideration of different systems and qubit error statistics. For surface and rotated surface structures [22], [26], considering i.i.d. data qubit errors, this step can be greatly simplified adopting the Manhattan distance between defects. Then, the MWPM proceeds by matching the graph to estimate the data qubit errors. A matching of a graph is a set of edges such that no two edges in the matching share a common vertex. A perfect matching is a matching that includes all vertices in the graph [27]. For topological quantum codes without boundaries (toric codes), we can directly search for perfect matches using the MWPM algorithm on the resulting graph [14]. For surface codes, which have boundaries, it is necessary to add ghost defects (also known as ghost ancillas [28]) before applying Dijkstra's algorithm. This is because errors occurring along a boundary excite only one ancilla, leading to an odd number of defects which impedes running the MWPM. Therefore, for each defect, a corresponding ghost defect is considered by the decoder. In the final graph, these defects are also connected between themselves with zero distance in order to preserve the error correction capability of the code [28].

The decoder based on the MWPM is able to guarantee the error correction capability t. In addition, it is also able to correct a significant amount of error patterns of weight greater than t. As an example, for the $[[9,1,3]]$ rotated surface code over the depolarizing channel, we have $\beta_2 = 0.5$, i.e., the decoder is able to correct 50% of the error patterns of weight two [26]. As implemented in the library for efficient modeling and optimization in networks (LEMON), the MWPM algorithm complexity is $O(NM \log N)$, where N and M represent the number of vertices and edges in the graph, respectively [29]. Specifically, for a toric code we have $N = n_{\mathrm{d}}$ and $M = \binom{n_{\mathrm{d}}}{2}$, while for a surface code $N = 2\,n_{\mathrm{d}}$ and $M = \binom{2\,n_{\mathrm{d}}}{2}$ due to ghost ancillas, where n_{d} denotes the number of defects.

Since planar architectures are favorable from an implementation point of view, we will focus on surface codes in the following. The STM decoder for toric codes can be easily constructed by straightforward variations on the algorithm described below.

III. Spanning Tree Matching Decoder

In this Section, we introduce the STM decoder for surface codes. First, from (1) we observe that, for a code with given n and t, the error correction capability is determined by the fraction of errors of weight $w = t+1$ that the decoder is able to correct. Since the surface codes belong to the class of Calderbank, Shor, and Steane (CSS) codes, for the sake of simplicity we will refer to the lattice where the sites constitute the $\boldsymbol{X}$ generators. The same reasoning can be applied to the dual lattice.

The STM decoder consists of three phases: MST evaluation, tree matching procedure, and error correction.

A. Minimum Spanning Tree Phase

We first construct the complete graph $\mathcal{G} = (n_{\mathrm{d}}, \binom{n_{\mathrm{d}}}{2})$ connecting all defects, by using Dijkstra or the Manhattan distances. Note that, since there are no ghost ancillas at this stage, this graph has fewer edges and vertices than the one used by the MWPM.

Then, we execute the MST algorithm on the graph. This can be achieved with a complexity of $O(M \log N)$, where $M = \binom{n_{\mathrm{d}}}{2}$ and $N = n_{\mathrm{d}}$ stand for the edges and the vertices of the graph, respectively [30]. At this stage, we construct two MSTs starting from the obtained one. If the number of defects is already even, one of the output trees is the original one, while the other is derived by introducing a ghost defect on both the left and right sides. If the number of defects is odd, an output tree is obtained by adding a ghost defect to the left, while the alternative output tree is built by adding a ghost defect to the right. Then, the added ghost defects are connected to the nearest defect in the lattice. An example of this phase is reported in Fig. 1(b). In case of multiple defects with a ghost defect at the same distance, we select the one with the highest minimum distance to other non-ghost defects.

B. Tree Matching Phase

During this step, we match the trees to obtain the estimated error patterns. For a tree $\mathcal{T}$ let us define as $\mathcal{E}$ its perfect matching graph. We use $\mathrm{adj}(v, \mathcal{G})$ to indicate the set of vertices adjacent to the vertex v in the graph $\mathcal{G}$, and $\deg(v, \mathcal{G})$ for the degree of v in the graph $\mathcal{G}$. $\mathcal{B}(\mathcal{T}) = \{v \in \mathcal{T} \,|\, \deg(v, \mathcal{T}) = 1\}$ is the set of boundary vertices in $\mathcal{T}$.

A simple algorithm for obtaining the perfect matching of an MST consists of the following steps.

1. For each $b \in \mathcal{B}(\mathcal{T})$

   Let $a = \mathrm{adj}(b, \mathcal{T})$;

   If $\deg(a, \mathcal{T}) = 2$: call the edges $e_1 = (a, b)$ and $e_2 = (a, \mathrm{adj}(a, \mathcal{T}) \backslash \{b\})$. Add e_1 to $\mathcal{E}$, and remove the subgraph $(\{a, b\}, \{e_1, e_2\})$ from $\mathcal{T}$.

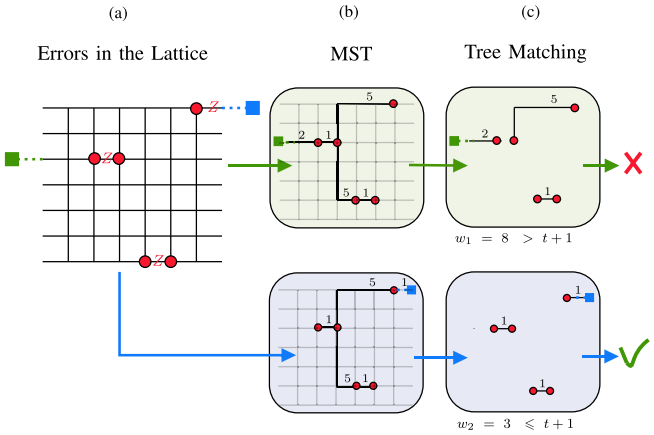

Fig. 1. Spanning tree matching decoder with a $[[85, 1, 7]]$ surface code. a) Three $\boldsymbol{Z}$ channel errors occur on the lattice. Exited ancillas are depicted in red. b) Two alternative MSTs obtained with the nearest ghost ancilla to the left (above) and to the right (below) boundary, respectively. c) Resulting $\mathcal{E}$ from the tree matching procedure.

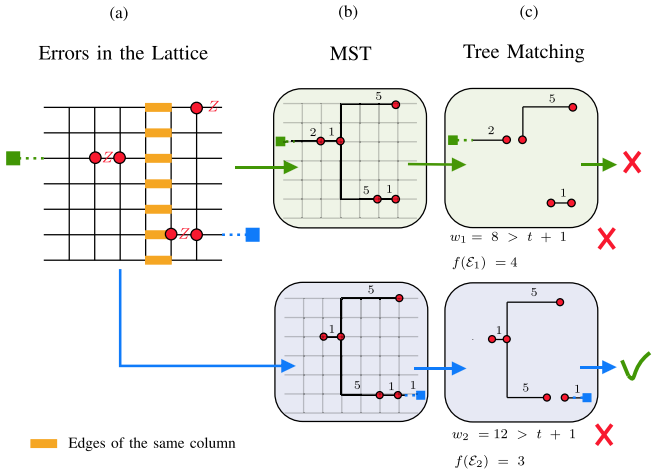

Fig. 2. Spanning tree matching decoder with a $[[85, 1, 7]]$ surface code. Both matched spanning trees have $w > t + 1$. Hence, the error correction is performed according to (2).

2. For each $b \in \mathcal{B}(\mathcal{T})$
   Let $a = \text{adj}(b, \mathcal{T})$;
   If $\deg(a, \mathcal{T}) = 3$: call $\{v_1, v_2\} = \text{adj}(a, \mathcal{T}) \setminus \{b\}$, and the edges $e_1 = (a, b)$, $e_2 = (a, v_1)$, $e_3 = (a, v_2)$. Add the edge e_1 to $\mathcal{E}$ and remove from $\mathcal{T}$ the subgraph $(\{a, b\}, \{e_1, e_2, e_3\})$. Insert the edge $e = (v_1, v_2)$ with weight $w(e) = w(e_2) + w(e_3)$ in $\mathcal{T}$. Return to step 1.
3. For each $b \in \mathcal{B}(\mathcal{T})$
   Let $a = \text{adj}(b, \mathcal{T})$;
   If $\deg(a, \mathcal{T}) = 4$: remove from $\mathcal{T}$ the edge $e = (a, v)$ where v is the sole vertex in $\text{adj}(a, \mathcal{T})$ with $\deg(v, \mathcal{T}) > 1$. Return to step 2.

Note that, considering i.i.d. data qubit errors, a vertex v in the MST cannot have $\deg(v, \mathcal{T}) > 4$.

The algorithm terminates when the graph $\mathcal{T}$ becomes empty, and gives a set of edges $\mathcal{E}$ representing a valid solution to the error correction problem. This procedure is carried out for both the MSTs obtained in Section III-A, and the two solutions are indicated as $\mathcal{E}_1$ and $\mathcal{E}_2$. Examples of the tree matching procedure described above are depicted in Fig. 1(c) and in Fig. 2(c).

C. Error Correction

After the previous phase we have two possible sets of faulty qubits, $\mathcal{E}_1$, $\mathcal{E}_2$, with a total number of data qubit errors w_1 and w_2, respectively. An instance of STM decoding, involving both MSTs, is illustrated in Fig. 1. If $w_1 \leq t + 1$ or $w_2 \leq t + 1$, then we have found the correction operator.[1]

If, instead, both w_1 and w_2 have weight $> t + 1$, a simple processing considering both $\mathcal{E}_1$ and $\mathcal{E}_2$ can be applied, which guarantees the correction if the channel errors has weight $\leq t$ (see Section III-D). The basic idea is choosing the solution with the smallest number of horizontally traversed edges. This is motivated by the fact that logical operators traverse the lattice from one side to the other one. Hence, the correction with more operators along the horizontal dimension will more likely cause a logical operator (i.e., an undetected error).

Definition 1: A column is the set of horizontal edges aligned in the vertical direction of the lattice, as shown in Fig. 2(a).

In this way, we can enumerate the columns from left to right ranging from 1 to d. Then, we can define two vectors $\boldsymbol{c}_i$ with entries $c_{i,j}$, where $i = 1, 2$ and $j = 1, \ldots, d$, representing the cardinality of the intersection between $\mathcal{E}_i$ and the j-th column. Since a solution $\mathcal{E}_i$ with two edges in the j-th column is equivalent, by adding a stabilizer, to another solution without any edge in the j-th column, we also define as $\boldsymbol{u}_i$ a vector with entries $u_{i,j} = c_{i,j} \mod 2$. As a consequence, the function

$$f(\mathcal{E}_i) = \sum_{j=1}^{d} u_{i,j} \tag{2}$$

can be used as a metric to quantify how much correcting $\mathcal{E}_i$ is likely to cause a logical operator. Thus, the final solution is that minimizing (2). For instance, in Fig. 2(c) there are three edges in solution $\mathcal{E}_1$, with weights $w = 1$, $w = 2$, and $w = 5$. They consist of one, two, and three horizontal qubits, respectively. However, the horizontal qubit of the edge with weight $w = 1$ belongs to the same column as the edge with weight $w = 5$. Hence, $f(\mathcal{E}_i) = 4$.

D. Distance-Preserving Decoding

In this section we show that (2), in the case of a $[[n, k, d = 2t + 1]]$ surface code, assures the correction of channel errors with weight $w \leqslant t$.

Lemma 1: Given a surface code, let us call $\mathcal{C}$ an arbitrary Pauli $\boldsymbol{Z}$ error chain connecting two sites defects v_1 and v_2. Then, the intersection of $\mathcal{C}$ with any column between v_1 and v_2 has cardinality $1 \pmod 2$. The intersection with any of the other columns has cardinality $0 \pmod 2$.

Proof: The minimal-weight chain linking v_1 and v_2 clearly satisfies the theorem. Every other chain can be obtained from this one, through the application of plaquette generators. Since each plaquette has two edges in the same column, it does not affect the modulo 2 counting. □

[1]Note that, if $w_1 \leqslant t$, the matching of the second tree can be avoided to save time.

Lemma 2: Given any error pattern over an $[[n, k, d]]$ surface code, resulting in an even number of site defects (including also ghost defects), all possible perfect matchings $\mathcal{M}$ have the same metric $f(\mathcal{M})$.

Proof: We label the horizontal position of the sites in the lattice, from left to right, as $0, 1, \ldots, d+1$, with 0 for the left ghost defect and $d+1$ for the right ghost defect. Vertically aligned sites share the same label. Let us firstly examine the case where the lattice has four site defects. Reordering them we obtain four indexes $0 \leq j_1 \leq j_2 \leq j_3 \leq j_4 \leq d+1$, representing the site positions. In this setup we can have three possible perfect matchings $\mathcal{M}_i$, with $i = 1, 2, 3$. Considering $\mathcal{M}_1$ as the one connecting site 1 with 2, and site 3 with 4, by application of Lemma 1, we have that $f(\mathcal{M}_1) = (j_2 - j_1) + (j_4 - j_3)$. It is easy to check that the other two matchings have the same metric. Finally, we observe that each perfect matching $\mathcal{M}$ in any graph $\mathcal{G}$ with an even number of defects can be obtained by iteratevely removing two edges and reconnecting them as needed. Hence, the claim follows since this operation does not affect the metric $f(\mathcal{M})$. □

Corollary 1: Each perfect matching has the same metric (2) as the MWPM. It follows that (2) can be interpreted as the minimum number of traversed columns for all perfect matchings of the same graph.

We have seen in Section III-A that, starting from a complete graph $\mathcal{G}$ formed by connecting all defects, we obtain two distinct graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ by adding ghost ancillas. Due to the even number of defects in these graphs, we can compute two perfect matchings, denoted as $\mathcal{E}_1$ and $\mathcal{E}_2$. It is noteworthy that applying a logical operator to one matching we get the other. Consequently, one of the solutions will correct the error (call it $\mathcal{E}_c$), while the alternative, $\mathcal{E}_a$, will produce a logical error.

Theorem 1: Let us consider an $[[n, k, d]]$ surface code and an error pattern of weight $w \leq t$. A perfect matching leading to the correct solution, $\mathcal{E}_c$, has $f(\mathcal{E}_c) < f(\mathcal{E}_a)$, with $\mathcal{E}_a$ being any perfect matching on the alternative graph.

Proof: The number of columns in the lattice is d. Since we are considering error patterns with weight $w \leq t$, we have that any perfect matching $\mathcal{E}_\mathrm{c}$ representing the correct solution, due to Lemma 2, satisfies $f(\mathcal{E}_\mathrm{c}) \leq t$. Regarding the alternative solution $\mathcal{E}_\mathrm{a}$, we have that $f(\mathcal{E}_\mathrm{a}) = d - f(\mathcal{E}_\mathrm{c})$, since they differ by a logical operator. This leads to $f(\mathcal{E}_\mathrm{a}) \geq t + 1$ for d odd and $f(\mathcal{E}_\mathrm{a}) \geq t + 2$ for d even, proving the statement. □

The previous theorem states that a surface decoder chosing a matching with the minimum traversed column metric (2) preserves the error correction capability of the code.

The same theorem applies to rotated surface codes as well, owing to their similar lattice structure.

IV. Rapid-Fire Decoder

We show now how to design an even faster decoder that ensures the correction for all errors with weights $w \leqslant t$. Due to Corollary 1, each matching is equivalent to the minimum one in terms of (2). Hence, it is possible to compute $\mathcal{E}_i$ with $i = 1, 2$ without evaluating and matching the MST. Firstly, we compute the complete graph $\mathcal{G}$ on the defects. Then, we construct two graphs $\mathcal{S}_i$ by adding to $\mathcal{G}$ ghost ancillas on boundaries with the same strategy described in Sections III-A. For each $\mathcal{S}_i$, we iteratively pair each defect with its closest one in a greedy fashion, we update the solution $\mathcal{E}_i$, and we remove both the defects from $\mathcal{S}_i$. This process results in two potential sets of faulty qubits, $\mathcal{E}_1$ and $\mathcal{E}_2$. Finally, we determine the error correction operator to adopt, following the procedure outlined in Section III-C. In this way, we guarantee the correction of all errors of weight up to $w = t$ by Theorem 1. We call this the RFire decoder.

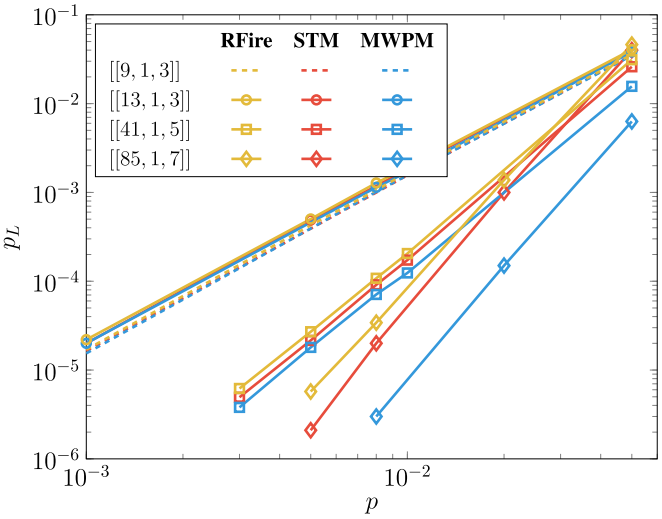

Fig. 3. Logical error probability vs. physical error probability of the channel. Rotated $[[9, 1, 3]]$, and standard $[[13, 1, 3]]$, $[[41, 1, 5]]$, and $[[85, 1, 7]]$ surface codes over depolarizing channel.

V. Numerical Results

In this section we compare the performance of surface and rotated surface codes using MWPM, STM, and RFire decoding via Monte Carlo simulations. These decoders are implemented in C++ and run with an Apple Silicon M2 processor. In the implementation, we exploit the LEMON C++ library for an efficient MWPM algorithm [29]. To evaluate the complexity of the decoders we measure the average execution time of the processing that starts with the initial complete graph $\mathcal{G}$ (generation of $\mathcal{G}$ not included) and ends returning the solution $\mathcal{E}$. The evaluation, carried out when varying the lattice size and the number of $\boldsymbol{Z}$ defects, is reported in Table I. As expected, the computational time saving is remarkable when STM and RFire are adopted. Furthermore, in Fig. 3, the logical error rate of some surface codes over a depolarizing channel is depicted. In particular, the performance gap between MWPM and the proposed fast decoders widens progressively as the distance of the code increases. This is due to the fact that, although the STM and the RFire decoders effectively correct all the errors with weights up to t, the MWPM decoder corrects a larger fraction of those with weight $t + 1$. In terms of speed, the RFire decoder is the fastest, but it looses in error correction capability. Note that although we showcased our decoder using standard surface codes for the sake of clarity, its performance remains consistent when applied to rotated

TABLE I
AVERAGE EXECUTION TIMES [μs]

		$n_d = 1$	$n_d = 2$	$n_d = 3$	$n_d = 4$	$n_d = 5$	$n_d = 6$	$n_d = 7$	$n_d = 8$
RFire		0.0008	0.0015	0.0021	0.0025				
STM	$[[13, 1, 3]]$	0.0025	0.0031	0.0051	0.0123				
MWPM		17.329	30.618	62.254	63.653				
RFire		0.0019	0.0044	0.0051	0.0081	0.0085	0.0108	0.0110	0.0341
STM	$[[85, 1, 7]]$	0.0031	0.0047	0.0091	0.0121	0.0695	0.2071	1.2839	1.9302
MWPM		19.156	32.247	55.864	67.591	100.55	115.04	157.67	171.78

surface types, as illustrated in Fig. 3 for the $[[9, 1, 3]]$ code. Among the decoders available in the literature, we concentrate on comparing with the only current fast decoder, which is the UF [10]. To accomplish this, we utilized the Qsurface library [10]. Specifically, for the $[[85, 1, 7]]$ surface code and a physical error rate $p_Z = 0.03$, the UF shows a speedup by a factor of approximately $\times 2$ with respect to MWPM. In the same settings, the speedup of RFire and MST decoders with respect to MWPM is $\times 3625$, and $\times 164$, resulting in the quickest execution times.

VI. CONCLUSION

We have introduced two fast decoders designed for surface codes and compared their performance with the widely used MWPM decoder. While observing a slight reduction in error correction capability for errors of weight $j \geqslant t+1$, our findings highlight a substantial advantage in terms of execution times, with a speedup of a factor $\times 10.000$, for both the STM and the RFire decoders.

REFERENCES

[1] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018.

[2] A. S. Cacciapuoti, M. Caleffi, R. Van Meter, and L. Hanzo, "When entanglement meets classical communications: Quantum teleportation for the quantum internet," *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3808–3833, Jun. 2020.

[3] F. Zaman, U. Khalid, T. Q. Duong, H. Shin, and M. Z. Win, "Quantum full-duplex communication," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 9, pp. 2966–2980, Sep. 2023.

[4] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A, Gen. Phys.*, vol. 52, no. 4, pp. R2493–R2496, Oct. 1995.

[5] D. Gottesman, "An introduction to quantum error correction and fault-tolerant quantum computation," in *Proc. Symp. Appl. Math.*, vol. 68, 2010, pp. 13–58.

[6] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2010.

[7] H. D. Pfister, C. Piveteau, J. M. Renes, and N. Rengaswamy, "Belief propagation for classical and quantum systems: Overview and recent results," *IEEE BITS Inf. Theory Mag.*, vol. 2, no. 3, pp. 20–32, Dec. 2023.

[8] F. Zoratti et al., "Improving the speed of variational quantum algorithms for quantum error correction," *Phys. Rev. A, Gen. Phys.*, vol. 108, no. 2, Aug. 2023, Art. no. 022611.

[9] S. B. Bravyi and A. Y. Kitaev, "Quantum codes on a lattice with boundary," 1998, *arXiv:quant-ph/9811052*.

[10] A. D. Iolius et al., "Decoding algorithms for surface codes," 2024, *arXiv:quant-ph/2307.14989*.

[11] O. Higgott, "PyMatching: A Python package for decoding quantum codes with minimum-weight perfect matching," *ACM Trans. Quantum Comput.*, vol. 3, no. 3, pp. 1–16, Sep. 2022.

[12] O. Higgott, T. C. Bohdanowicz, A. Kubica, S. T. Flammia, and E. T. Campbell, "Improved decoding of circuit noise and fragile boundaries of tailored surface codes," *Phys. Rev. X*, vol. 13, no. 3, Jul. 2023, Art. no. 031007.

[13] B. J. Brown, "Conservation laws and quantum error correction: Towards a generalised matching decoder," *IEEE BITS Inf. Theory Mag.*, pp. 1–12, 2023.

[14] V. Kolmogorov, "Blossom V: A new implementation of a minimum cost perfect matching algorithm," *Math. Program. Comput.*, vol. 1, no. 1, pp. 43–67, Jul. 2009.

[15] B. M. Terhal, "Quantum error correction for quantum memories," *Rev. Mod. Phys.*, vol. 87, no. 2, pp. 307–346, Apr. 2015.

[16] A. G. Fowler, "Minimum weight perfect matching of fault-tolerant topological quantum error correction in average $O(1)$ parallel time," 2013, *arXiv:1307.1740*.

[17] L. Skoric, D. E. Browne, K. M. Barnes, N. I. Gillespie, and E. T. Campbell, "Parallel window decoding enables scalable fault tolerant quantum computation," *Nature Commun.*, vol. 14, no. 1, p. 7040, Nov. 2023.

[18] N. Delfosse and G. Zémor, "Linear-time maximum likelihood decoding of surface codes over the quantum erasure channel," *Phys. Rev. Res.*, vol. 2, no. 3, Jul. 2020, Art. no. 033042.

[19] N. Delfosse and N. H. Nickerson, "Almost-linear time decoding algorithm for topological codes," *Quantum*, vol. 5, p. 595, Dec. 2021.

[20] S. Gicev, L. C. L. Hollenberg, and M. Usman, "A scalable and fast artificial neural network syndrome decoder for surface codes," *Quantum*, vol. 7, p. 1058, Jul. 2023.

[21] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory," *J. Math. Phys.*, vol. 43, no. 9, pp. 4452–4505, Sep. 2002.

[22] D. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, "Surface code quantum computing by lattice surgery," *New J. Phys.*, vol. 14, no. 12, Dec. 2012, Art. no. 123011.

[23] J. P. B. Ataides, D. K. Tuckett, S. D. Bartlett, S. T. Flammia, and B. J. Brown, "The XZZX surface code," *Nature Commun.*, vol. 12, no. 1, p. 2172, Apr. 2021.

[24] D. Bluvstein et al., "Logical quantum processor based on reconfigurable atom arrays," *Nature*, vol. 626, no. 7997, pp. 58–65, Feb. 2024.

[25] D. Forlivesi, L. Valentini, and M. Chiani, "Performance analysis of quantum CSS error-correcting codes via MacWilliams identities," 2023, *arXiv:2305.01301*.

[26] D. Forlivesi, L. Valentini, and M. Chiani, "Logical error rates of XZZX and rotated quantum surface codes," *IEEE J. Sel. Areas Commun.*, early access, Mar. 21, 2024, doi: 10.1109/JSAC.2024.3380088.

[27] R. Diestel, A. Schrijver, and P. Seymour, "Graph theory," *Oberwolfach Rep.*, vol. 4, no. 2, pp. 887–944, 2008.

[28] S. Hadfield, "Error rate thresholds for physical implementations of the surface code," Ph.D. dissertation, Dept. School Physics, Univ. Melbourne, Melbourne, VIC, Australia, 2008.

[29] B. Dezső, A. Jüttner, and P. Kovács, "LEMON—An open source C++ graph template library," *Electron. Notes Theor. Comput. Sci.*, vol. 264, no. 5, pp. 23–45, Jul. 2011.

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

Open Access funding provided by 'Alma Mater Studiorum - Università di Bologna' within the CRUI CARE Agreement