Enhanced Message-Passing Decoding of Degenerate Quantum Codes Utilizing Trapping Set Dynamics

Dimitris Chytas, Michele Pacenti, *Graduate Student Member, IEEE,* Nithin Raveendran, *Member, IEEE,* Mark F. Flanagan, *Senior Member, IEEE,* and Bane Vasić, *Fellow, IEEE*

***Abstract*—In this letter, we propose a novel iterative decoding algorithm that exploits the degenerate nature of three different families of quantum low-density parity-check codes, i.e., surface, toric, and row-degree-4 bicycle codes. Such families of codes share harmful trapping sets that constitute symmetric stabilizers, making it impossible for any parallel-scheduled iterative message-passing decoder to converge even for error patterns of weight as low as two. By biasing subsets of nodes in the symmetric stabilizers, the decoder is able to converge to a valid error pattern. Furthermore, the proposed decoder has low decoding complexity - linear in the code's blocklength - and a fully parallel schedule, making it suitable for low-latency efficient implementation.**

***Index Terms*—QLDPC codes, belief propagation decoding, topological codes, degeneracy, symmetric stabilizers.**

I. Introduction

QUANTUM states are intrinsically fragile, and quantum error correction is needed for realizing fault-tolerant quantum information processing. A stabilizer code can be used to protect information stored in a quantum state, such that measurement of the syndrome is used to detect and correct errors without disturbing the information stored in that state [1]. Many quantum stabilizer codes, such as topological codes, can be represented using sparse graphs; in this letter, we consider surface and toric codes [2] and bicycle codes [3].

The quantum decoding problem is finding the most probable coset of degenerate errors with a given error syndrome. Although message-passing iterative algorithms, such as belief propagation (BP), work extremely well for classical error-correcting codes with almost linear complexity, it is well known that, in principle, their performance on quantum codes remains poor. This is true, especially for degenerate codes that contain many low-weight stabilizers compared to their minimum distance [4]. The best-performing decoding algorithms for quantum codes are belief propagation with ordered statistics decoding (BP-OSD) [5] and minimum weight perfect matching (MWPM) [6] for topological codes. However, their high complexity makes them unsuitable for large-scale implementation. There have been many attempts to design improved message-passing decoders for quantum codes, e.g., in [7] Poulin and Chung proposed heuristic modifications to BP, called *freezing*, *random perturbation* and *collision*, which improved the performance of highly degenerate codes. However, these approaches are not deterministic and do not fully exploit the code structure. Also, in [8] and [9], variants of BP are shown to provide good performance if paired with serial scheduling; nevertheless, serial scheduling is poorly matched to the tight latency requirements of fault-tolerant quantum systems. The decoding approach of [9], in contrast to other BP-based approaches, modifies the variable-to-check update rule by optimizing the "inhibition" term, which allows the decoder to avoid propagation of incorrect beliefs in short cycles. Targeting topological codes, the authors in [10] proposed branch-assisted sign-flipping belief propagation (BSFBP) decoding. Their approach improves BP by introducing new decoding paths branched from BP combined with a syndrome residual obtained from the syndrome-pruning process. However, significant improvements are achieved only when BSFBP is combined with OSD, thus not solving the latency issue.

The main problem that prevents BP from performing well on quantum codes is that of *error degeneracy*, i.e., given a syndrome, there can be multiple equally likely error patterns that can match it. This phenomenon can be studied considering harmful structures in the code's graph called *trapping sets (TSs)*.

In this letter, we propose a novel decoder called *BP with bias using oscillating trapping sets (BP-OTS)*, that retains the critical advantages of BP such as low complexity ($\mathcal{O}(n)$) and low latency due to a fully parallel schedule, and is also able to escape trapping sets, thus achieving much better decoding performance than BP.

The letter is organized as follows: Section II provides the preliminaries regarding quantum error correcting codes and decoders. In Section III, we present the algorithm and explain the rationale behind the proposed decoder. In Section IV, we provide an analysis of the oscillatory dynamics of a TS and how this can be exploited while decoding. Section VI provides decoding results of BP-OTS versus BP and the other BP-based decoders mentioned above. Finally, conclusions and future research directions are provided in Section VII.

II. Preliminaries

Consider the n-fold Pauli group,

$$\mathcal{G}_n \triangleq \{cB_1 \otimes \cdots \otimes cB_n : c \in \{\pm 1, \pm i\}, B_j \in \{I, X, Y, Z\}\},$$

where I, X, Y and Z are called *Pauli operators*. Every element in $\mathcal{G}_n$ has eigenvalues ± 1, and any two elements in $\mathcal{G}_n$ either

This work has been conducted as a part of the CoQREATE program, which is funded by the National Science Foundation under grant ERC-1941583 and Science Foundation Ireland under the US-Ireland R&D Partnership Programme under grant SFI/21/US-C2C/3750. The work of Dimitris Chytas, Michele Pacenti, Nithin Raveendran and Bane Vasić is also supported by the NSF under grants CIF-1855879, CIF-2106189, CCF-2100013, ECCS/CCSS-2027844, ECCS/CCSS-2052751, and in part by Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration and funded through JPL's Strategic University Research Partnerships (SURP) program. Bane Vasić has disclosed an outside interest in his startup company, Codelucida to The University of Arizona. Conflicts of interest resulting from this interest are being managed by The University of Arizona in accordance with its policies.

Dimitris Chytas, Michele Pacenti, Nithin Raveendran and Bane Vasić are with the Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ 85721, USA (e-mail: dchytas@arizona.edu; mpacenti@arizona.edu; nithin@arizona.edu; vasic@ece.arizona.edu).

Mark F. Flanagan is with the School of Electrical and Electronic Engineering, University College Dublin, Belfield, Dublin 4, Ireland (e-mail: mark.flanagan@ieee.org).

commute or anticommute with each other. A stabilizer group $\mathcal{S}$ is an Abelian subgroup in $\mathcal{G}_n$. If $\mathcal{S}$ is generated by $n-k$ independent generators, it defines a $[\![n,k,d]\!]$ stabilizer code $\mathcal{C}$ that encodes k logical qubits into n physical qubits, with d being its minimum distance. The elements of $\mathcal{S}$ are called stabilizers. The set of generators of $\mathcal{S}$ can be represented by the stabilizer matrix $\mathbf{H}$, whose $(i,j)^{\text{th}}$ element is given by the Pauli operator corresponding to the j^{th} qubit in the i^{th} stabilizer.

We consider a depolarizing channel characterized by channel parameter ϵ (depolarizing error rate), which induces error pattern $\mathbf{e} \in \{I,X,Z,Y\}^n$ on the n qubits. In the depolarizing channel, every qubit goes through either a bit-flip (X), phase-flip (Z), or both (Y), each with probability $\epsilon/3$. The probability of no error (I) on the same qubit is then equal to $1-\epsilon$. The weight of $\mathbf{e}$ is defined as the number of non-identity operators in $\mathbf{e}$. Unlike classical error correction, the decoder cannot estimate the codeword without perturbing the codeword state. This issue can be circumvented by measuring the syndrome. In this letter, we assume a noise-free syndrome measurement.

Given the binary syndrome vector $\mathbf{s}$ and the stabilizer matrix $\mathbf{H}$, a quantum error-correcting decoder estimates the most likely error pattern $\hat{\mathbf{e}} \in \{I,X,Z,Y\}^n$. The decoding process succeeds when the estimated error pattern $\hat{\mathbf{e}}$ is equal to the actual error pattern, or one of the equivalent error patterns that lead to the same syndrome (this is known as *error degeneracy*); otherwise a logical error occurs. Based on the Pauli-to-binary isomorphism [11], $\mathbf{e}$ can be modeled as a binary vector of length $2n$, i.e., $\mathbf{e} = (\mathbf{e}_X, \mathbf{e}_Z)$. Both $\mathbf{e}_X$ and $\mathbf{e}_Z$ are length-n binary vectors representing X and Z errors, respectively. For instance, an X error on the j^{th} qubit will be represented by 1 and 0 at the j^{th} and $(n+j)^{\text{th}}$ index of $\mathbf{e}$, respectively.

In this letter, we address topological and bicycle codes, which constitute classes of Calderbank-Shor-Steane (CSS) codes. A $[\![n, k_X - k_Z, d]\!]$ CSS code [12] is a stabilizer code constructed using two classical codes, $C_X[\![n,k_X,d_X]\!]$ and $C_Z[\![n,k_Z,d_Z]\!]$, where $d \geq \min\{d_X, d_Z\}$ and $C_Z \subset C_X$. Given the stabilizer matrix $\mathbf{H}_X$ of C_X and $\mathbf{H}_Z$ of $C_Z^{\perp}$, the CSS code constructed from C_X and C_Z has the form $\mathbf{H} = \left[\begin{smallmatrix} \mathbf{H}_Z & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_X \end{smallmatrix}\right]$, where $\mathbf{H}_X \cdot \mathbf{H}_Z^T = \mathbf{0}$ (unless specified otherwise, we assume all operations on binary matrices and vectors are performed on the binary field). CSS codes facilitate binary decoding because of their structure, i.e., X errors are decoded using $\mathbf{H}_Z$, and Z errors are decoded using $\mathbf{H}_X$. The corresponding input syndromes are obtained as $\mathbf{s}_X = \mathbf{e}_X \cdot \mathbf{H}_Z^T$ and $\mathbf{s}_Z = \mathbf{e}_Z \cdot \mathbf{H}_X^T$, respectively. The $\mathbf{H}_X$ and $\mathbf{H}_Z$ stabilizer matrices can each be represented as a *Tanner graph* $\mathcal{T} = \mathcal{T}(V, C, \mathcal{E})$, which is a bipartite graph where $V = \{v_1, ..., v_n\}$ is the set of n variable (qubit) nodes, $C = \{c_1, ..., c_m\}$ is the set of m check nodes, and $\mathcal{E}$ is the set of edges (non-identity entries of the stabilizer matrix) connecting them. We denote by $\mathcal{M}(v_j)$ the indices of the neighboring check nodes of v_j, and by $\mathcal{N}(c_j)$ the indices of the neighboring variable nodes of the check node c_i. For the rest of this letter, only one type of error (X or Z) and its decoding will be considered; without loss of generality, the notation $\mathbf{H}$ will refer to $\mathbf{H}_Z$, $\mathbf{e}$ will refer to $\mathbf{e}_X$, and $\mathbf{s}$ will refer to $\mathbf{s}_X$.

A message-passing decoder is an iterative algorithm that operates on the Tanner graph of the code. Variable-to-check messages $\nu_{j\to i}^{(\ell)}$ and check-to-variable messages $\mu_{i\to j}^{(\ell)}$ are computed in each iteration ℓ at variable and check nodes respectively, as functions of the incoming messages, and propagated via the edges of the graph. The output of the decoder is an estimate of $\Pr(\mathbf{e}|\mathbf{s})$, i.e., the conditional probability of the error $\mathbf{e}$ given the observed syndrome $\mathbf{s}$. For this letter, we consider BP as the reference message-passing decoder whose performance we wish to improve.

III. BP WITH BIAS USING OSCILLATING TRAPPING SETS

In this section, we describe BP-OTS, an iterative message-passing decoder which is also illustrated in Algorithm 1. Variable-to-check and check-to-variable messages are computed using the following equations, respectively:

$$\nu_{j\to i}^{(\ell)} = \Omega_j + \sum_{i' \in \mathcal{M}(v_j)\setminus\{c_i\}} \mu_{i'\to j}^{(\ell)}, \tag{1}$$

and

$$\mu_{i\to j}^{(\ell)} = (-1)^{s_i} \cdot 2\tanh^{-1}\left\{ \prod_{j' \in \mathcal{N}(c_i)\setminus\{v_j\}} \tanh\left\{\frac{1}{2}\nu_{j'\to i}^{(\ell-1)}\right\}\right\}, \tag{2}$$

where Ω_j is a real number defined as

$$\Omega_j = \begin{cases} \Pi_j, \text{ if } v_j \text{ is not biased} \\ -C, \text{if } v_j \text{ is biased} \end{cases} \tag{3}$$

with $\Pi_j = \log\left\{\frac{1-(2\epsilon/3)}{2\epsilon/3}\right\}$ being the *a priori* log-likelihood-ratio value for v_j, and C a positive constant whose value needs to be designed. We will introduce the notion of *biasing* later in this section, as none of the variable nodes are biased in the first $T-1$ iterations which are equivalent to the classical BP iterations. After the computation of the messages, the *a posteriori* log-likelihood ratio (LLR) for the j^{th} qubit at the ℓ^{th} iteration is computed using the following equation:

$$q_j^{(\ell)} = \Omega_j + \sum_{i' \in \mathcal{M}(v_j)} \mu_{i'\to j}^{(\ell)}. \tag{4}$$

Then, the hard decision estimate $\hat{e}_j^{(\ell)}$ for the j^{th} qubit is computed as follows:

$$\hat{e}_j^{(\ell)} = \begin{cases} 0, \text{ if } q_j^{(\ell)} \geq 0 \\ 1, \text{ if } q_j^{(\ell)} < 0 \end{cases}. \tag{5}$$

In addition, for each variable node, we count how many times its hard decision estimate $\hat{e}_j^{(\ell)}$ changes during the $T-1$ iterations; this can be done by summing (modulo-2) $\hat{e}_j^{(\ell)}$ and $\hat{e}_j^{(\ell-1)}$ for each variable node. If $\hat{e}_j^{(\ell)} = \hat{e}_j^{(\ell-1)}$, their sum will be zero, meaning that the hard decision estimate remained the same in the two iterations; vice versa, if $\hat{e}_j^{(\ell)} \neq \hat{e}_j^{(\ell-1)}$, their sum will be one, meaning that the hard decision estimate changed in the two iterations. This leads us to introduce the *oscillation vector* $\boldsymbol{\sigma}$, a counter that increments every time the hard decision of a variable node changes during the iterations. These operations are summarized in lines 8-12 of Algorithm 1.

The *biasing step* is triggered every T iterations, where T is the biasing period, based on two criteria. First, we define $\mathcal{F} = \arg\max_{j\in[n]} \sigma_j$ to be the indices of those variable nodes with maximum oscillations. Let j_1 be the index of a variable node with maximum oscillations and minimum *a posteriori* LLR such that $j_1 = \arg\min_{j\in\mathcal{F}} |q_j^{(\ell)}|$. We choose to *bias* that particular node by setting $\Omega_{j_1} = -C$. Additionally, we reset its oscillations counter σ_{j_1} to zero to prevent it from being biased again in

This article has been accepted for publication in IEEE Communications Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2024.3356312

subsequent iterations. Secondly, we also choose to bias another variable node j_2, which has the minimum $|q_j^{(\ell)}|$ among all the nodes (note that j_1 and j_2 may coincide). The procedure is summarized in lines 14-23 of Algorithm 1. Biasing is performed recursively once every T iterations and is followed by $T-1$ standard BP iterations until the syndrome has been matched, or the maximum number of iterations has been reached.

A. Motivation

In classical error correction, BP (like any message-passing decoder) fails to correct certain error patterns when they appear inside particular structures of the Tanner graph. These harmful structures are referred to as *Trapping Sets* (TSs) and are defined as follows:

Definition 1. *[13] A trapping set for a syndrome-based iterative decoder is a non-empty set of variable nodes in a Tanner graph $\mathcal{G}$ that are not eventually converged or are neighbors of the check nodes that are not eventually satisfied.*

If the sub-graph induced by such a set of variable nodes has a variable nodes and b unsatisfied (odd degree) check nodes, then the TS is referred to as an (a, b) TS [13].

The structure of surface and toric codes [14] is given by the well-known lattice representation, where edges are qubits, vertices are X checks, and faces are Z checks (or vice versa). The Tanner graphs of the X and Z stabilizers reflect the same structure, with the difference that edges are replaced by variable nodes, meaning that Tanner graphs of surface codes are composed of a series of juxtaposed 8-cycles (i.e., closed loops formed by four variable nodes and four check nodes) arranged to form a lattice. Note that row-degree-4 bicycle codes [3] share the same type of Tanner graph. This symmetric structure prevents BP from correcting any two erroneous variable nodes inside an 8-cycle because of error degeneracy (shown in Fig. 1). Therefore, from now on, we will refer to 8-cycles in the aforementioned codes as $(4, 0)$ TSs; note that in literature, these are also known as symmetric stabilizers [13].

Notice that the $(4, 0)$ TSs (or 8-cycles) correspond to single tiles in the lattice graph of the code. Joining two adjacent $(4, 0)$ TSs forms a $(7, 2)$ TS. Generally, bigger TSs are obtained by combining adjacent $(4, 0)$ TSs, which can then be considered the building block of every TS of the code. It is then natural to think that a decoder able to deal with $(4, 0)$ TSs would also be able to deal with larger structures.

We observed that when a weight-2 error pattern lies inside a $(4, 0)$ TS, the decoder's hard decision $\hat{\boldsymbol{e}}$ oscillates back and forth between all four variable nodes of TS and the all-zero vector. Decoding can be, in principle, improved if we can detect such oscillating variable nodes. This observation led us to define the length-n *oscillation vector* $\boldsymbol{\sigma}$ which can detect the variable nodes belonging to problematic TSs.

After the highest oscillating variable nodes have been detected by $\boldsymbol{\sigma}$, the method of variable node *biasing* is utilized. The highest oscillating variable node (with minimum *a posteriori* LLR) j_1 is assumed to be erroneous by the decoder. In other words, in (1) and (4), we modify the *a priori* information associated to j_1 to bias the decoder towards estimating that node as erroneous. In this way, BP can converge to one of the degenerate pairs in the $(4, 0)$ TS due to the induced asymmetry. Note that (2) remains the same as BP's standard check to variable update rule. As mentioned before, the variable node j_2 is also biased. We observed that by biasing two nodes, BP-OTS can converge faster to degenerate error patterns.

Our approach is comparable with that in [7], where the authors propose the *freezing* technique, which consists of fixing the *a priori* LLR of a random variable node connected to an unsatisfied check. However, the method of [7] does not exploit the structure of the code and might result in freezing variable nodes that are not involved in any error pattern. In contrast, our proposed approach detects a subset of variable nodes such that they must be involved in one of the degenerate error patterns (as they belong to some TS); therefore, biasing one (or more) of identified variable nodes always helps the decoder to converge.

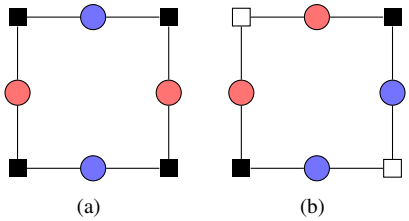

Fig. 1: The two uncorrectable weight-2 error patterns inside the $(4, 0)$ TS, observed in toric codes, surface codes and row-degree-4 bicycle codes. The red and blue colored circles (variable nodes) correspond to the equivalent error patterns. Black squares indicate unsatisfied checks, while white square nodes correspond to satisfied checks.

IV. Decoding of weight-2 errors

In this section, we analyze the oscillatory dynamics of the $(4, 0)$ TS and show how it can be exploited to correct weight-2 degenerate errors.

Assume we have a weight-2 error pattern. If this is not a degenerate error, i.e., both erroneous nodes belong to different $(4, 0)$ TSs, BP will correct it before reaching the T^{th} iteration (here, we assume that T is carefully designed not to let the decoder bias too early; typically, $T = 9$ is a good choice). We have observed that, for either of the error patterns shown in Fig. 1 (a), the oscillation vector for the variable nodes belonging to the TS is always equal to T (the sign of their LLRs changes at each iteration), while for the error patterns of Fig. 1 (b), the oscillation vector of the nodes will always be equal to $T/2$ (the sign of their LLRs changes every other iteration); on the other hand, the oscillation vector for any variable node outside the TS is always equal to 0. This can be intuitively explained by simply looking at the sign of the messages propagating in the TS. Consider two variable nodes v_{j_1}, v_{j_2} belonging to a $(4, 0)$ TS, and the check nodes c_{i_1}, c_{i_2}, such that c_{i_1} and c_{i_2} are neighbors of v_{j_1} and only c_{i_2} is also neighbor of v_{j_2}. Because of the structure of the code (and assuming all the incoming variable-to-check messages from outside the TS are positive), we have $\operatorname{sgn}\left(\mu_{i_2\to j_2}^{(\ell)}\right) = (-1)^{s_{i_2}} \cdot \operatorname{sgn}\left(\mu_{i_1\to j_1}^{(\ell-1)}\right)$, where $\operatorname{sgn}(.)$ is the sign function. Moreover, since variable nodes have degree 2, we have $\nu_{j_1\to i_2}^{(\ell)} = \mu_{i_1\to j_1}^{(\ell-1)}$. Since the amplitudes of the messages are all equal (due to the symmetry of the Tanner graph), there is a change in the sign of the LLR of a variable node only when

Algorithm 1 BP with bias using Oscillating Trapping Sets (BP-OTS)

Input: ϵ, $\mathbf{s}$, $\mathbf{H}$, T, L
Output: $\hat{\mathbf{e}}$

$\ell \leftarrow 1$ ▷ Initialization
$\Pi_j \leftarrow \log\left\{\frac{1-(2\epsilon/3)}{2\epsilon/3}\right\}$, $\forall j \in [n]$
$\Omega_j \leftarrow \Pi_j$, $\forall j \in [n]$
$\hat{e}_j^{(1)} \leftarrow 0$, $\forall j \in [n]$
$\sigma_j \leftarrow 0$ $\forall j \in [n]$
$\mu_{i\to j}^{(1)} \leftarrow 0$ $\forall\ i \in [m],\ j \in [n]$
while $\ell \leq L$ **do**
  Variable node update (1)
  Check node update (2)
  A posteriori log-likelihood computation (4)
  Hard decision (5)
  $\sigma_j \leftarrow \sigma_j + \left(\hat{e}_j^{(\ell)} \oplus \hat{e}_j^{(\ell-1)}\right)$ ▷ OVU
  if $\hat{\mathbf{e}} \cdot \mathbf{H}^T = \mathbf{s}$ **then return** $\hat{\mathbf{e}}$
  else if $\ell = kT$, $k \in \mathbb{N}$ **then**
    $\Omega_j \leftarrow \Pi_j,\ \forall j \in [n]$
    if $\max(\boldsymbol{\sigma}) > 0$ **then**
      $\mathcal{F} \leftarrow \arg\max_{j\in[n]} \sigma_j$
      $j_1 \leftarrow \arg\min_{j\in\mathcal{F}} |q_j^{(\ell)}|$
      $\sigma_{j_1} \leftarrow 0$ ▷ Reset oscillations
    end if
    $j_2 \leftarrow \arg\min_{j\in[n]} |q_j^{(\ell)}|$
    $\Omega_{j_1}, \Omega_{j_2} \leftarrow -C$
  end if
end while

both the incoming messages have the same sign. Fig. 2 depicts one period of these dynamics, with reference to either of the error patterns in Fig. 1 (a). The evolution of the hard decision on the variable nodes ultimately impacts the oscillation vector $\boldsymbol{\sigma}$, making the decoder aware of the TS. Fig. 3 demonstrates the effect of biasing a variable node on the messages in the TS and how it leads to correcting the error. A similar analysis can be carried out for the error patterns in Fig. 1 (b).

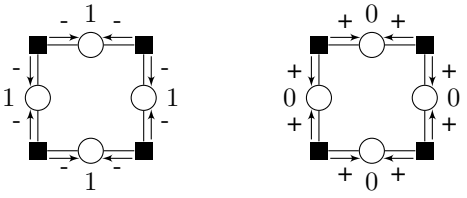

Fig. 2: Evolution (left to right) in two decoding iterations of the signs of check-to-variable messages and of the hard decision on the variable nodes for the error in Fig. 1 (a).

V. Complexity analysis

The standard iteration of the BP-OTS decoder is a BP iteration and thus involves computing nd_v variable-to-check messages and md_c check-to-variable messages, where d_v and d_c are the variable and check node degrees of the code, respectively. We assume that updating the oscillation vector $\boldsymbol{\sigma}$ at every iteration can be done in parallel for each variable node, with a complexity of $\mathcal{O}(1)$. For every T standard iterations, the biasing procedure is performed. It involves resetting the prior LLRs for each variable node, obtaining the indices j_1 and j_2, and changing the value of their *a priori* LLRs with biased values. These operations can also be easily implemented in parallel and have a negligible contribution to the overall complexity. Therefore, in the asymptotic regime, our decoder has a complexity comparable with the complexity of BP, i.e., linear in the code's blocklength. On the other hand, the complexity of MWPM is proportional to $\mathcal{O}(|\mathbf{s}|^3)$, where $|\mathbf{s}|$ is the Hamming weight of the syndrome, and the complexity of BP-OSD is proportional to $\mathcal{O}(n^3)$. The decoders in [8], [9] also have linear complexity, but they use a serial schedule in contrast to the flooding schedule used by BP-OTS; thus, the latency of the proposed algorithm is significantly lower. Finally, the decoder in [10] achieves significative performance only when paired with OSD, so its overall complexity is proportional to $\mathcal{O}(n^3)$.

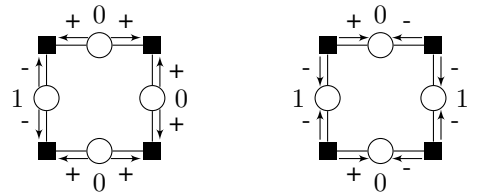

Fig. 3: Decoding iteration after biasing the leftmost variable node for the case of Fig 2. On the left, the signs of variable-to-check messages are illustrated; on the right, there are signs of subsequent check-to-variable messages.

VI. Performance evaluation

This section presents simulation results for the proposed BP-OTS algorithm and compares its performance with that of BP, MBP4 [9], and BSFBP [10] algorithms for surface, toric, and row-degree-4 bicycle codes. Each data point in the figures is obtained by simulating the corresponding decoder until we observe 100 logical errors. BP-OTS runs for a maximum of 200 iterations under parallel scheduling unless stated otherwise. To choose the optimal period T for each code, we fix the channel parameter and obtain the logical error rate performance across multiple periods $T \in \{2, 3, \cdots, 10\}$. We then choose the value of T that minimizes the logical error rate.

Fig. 4 demonstrates that BP-OTS can significantly improve the logical error rate performance of surface codes compared to BP. The main reason behind the poor performance of BP is the presence of $(4, 0)$ (and larger) TS in the Tanner graph; somewhat counter-intuitively, these TSs lead to a decrease in the code's error-correcting performance even as the code's minimum distance increases. We also compare our decoder with MBP4 for surface codes of minimum distance $d = 5, 7$, and 9; it can be observed that BP-OTS outperforms MBP4 for the cases of $d = 5$ and $d = 7$, and approaches its performance for $d = 9$; it is important to emphasize here that our decoder implements a fully parallel schedule, while MBP4 uses a serial message-passing schedule, meaning that BP-OTS introduce a significantly smaller decoding latency than MBP4.

In Fig. 5, we compare the performance of our scheme versus that of BSFBP and BP for toric codes with increasing minimum distance. The BSFBP decoder proposed in [10] also uses a parallel schedule. To be consistent with the work in [10], we only consider bit-flip (X) errors applied through a binary symmetric channel with crossover probability p. We observe

This article has been accepted for publication in IEEE Communications Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2024.3356312

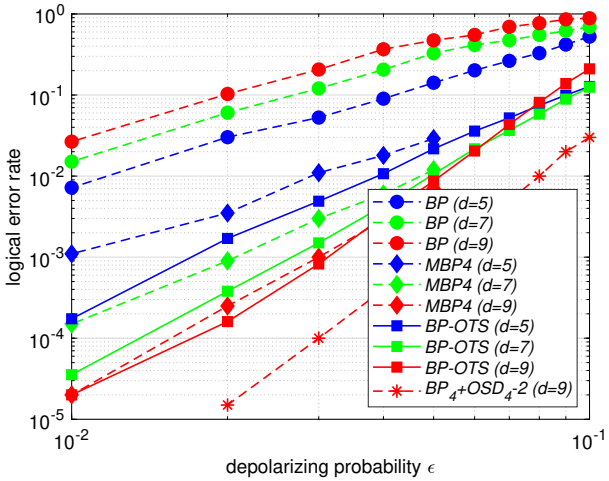

Fig. 4: Performance of BP-OTS versus BP and serial MBP4 for surface codes. The biasing period is set to $T = 9$. The plot for quaternary BP and OSD is replicated based on [15].

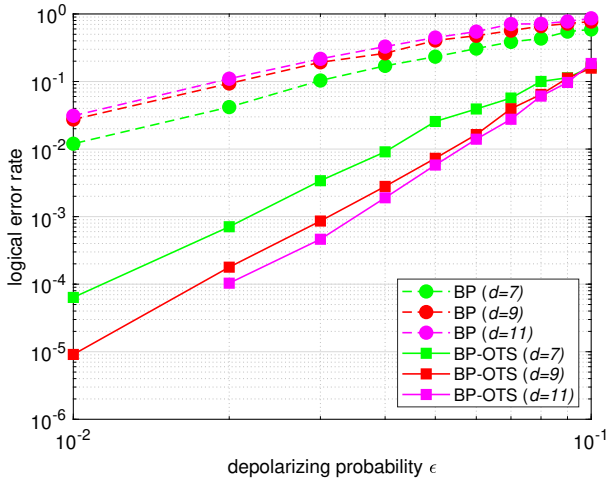

Fig. 6: Performance of BP-OTS versus BP for $[\![58, 2, 7]\!]$, $[\![106, 2, 9]\!]$ and $[\![134, 2, 11]\!]$ bicycle codes. The biasing period is set to $T = 7$.

that the proposed decoder significantly outperforms both BP and BSFBP. Indeed, we can perform better on the distance 5 surface code than BSFBP on the distance 9 surface code, while BP-OTS on the distance 9 surface code performs significantly better. As claimed in [10], BSFBP is able to correct a fraction of weight-2 and weight-3 error patterns, which are uncorrectable by BP; however, in our simulations, we observed that BP-OTS never failed to correct any error pattern of weight up to 4, which explains its performance gain compared to BSFBP.

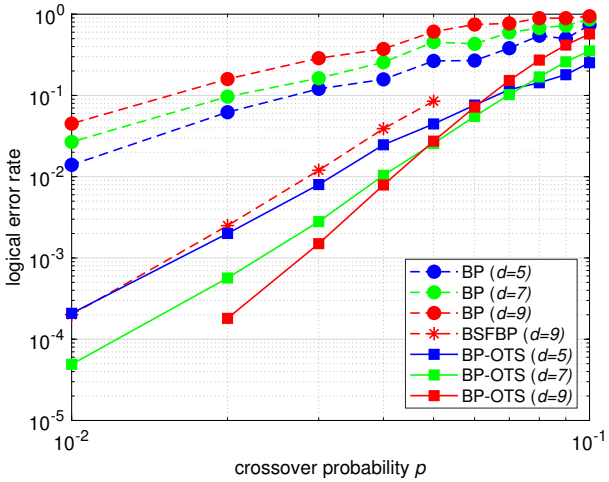

Fig. 5: Performance of BP-OTS versus BP and BSFBP for toric codes. The biasing period is set to $T = 9$. The plot for BSFBP was obtained from [10].

Finally, Fig. 6 demonstrates that BP-OTS significantly outperforms BP for various degree-4 bicycle codes as they share a similar structure with surface and toric codes.

VII. Conclusions and Future Work

We have proposed a new decoding scheme applicable to topological codes and certain families of bicycle codes. Our algorithm is able to deal with degeneracy by recognizing the trapping sets responsible for the decoding failure and biasing specific nodes in them to force the decoder to target one among many possible degenerate error patterns. Our proposed decoder has low complexity and a fully parallel schedule and outperforms other BP-based decoding schemes in the literature. Future work will involve the systematic study of degenerate error patterns and trapping sets for these classes of codes, with the goal of further improving our decoder.

References

[1] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A*, vol. 52, pp. R2493–R2496, Oct. 1995.
[2] A. Kitaev, "Fault-tolerant quantum computation by anyons," *Annals of Physics*, vol. 303, no. 1, pp. 2 – 30, 2003.
[3] D. J. C. MacKay, G. Mitchison, and P. L. McFadden, "Sparse-graph codes for quantum error correction," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2315–2330, Oct. 2004.
[4] P. Fuentes, J. Etxezarreta, P. Crespo, and J. Garcia-Frias, "Degeneracy and its impact on the decoding of sparse quantum codes," *IEEE Access*, vol. 9, pp. 89 093–89 119, Jun. 2021.
[5] P. Panteleev and G. Kalachev, "Degenerate quantum LDPC codes with good finite length performance," *Quantum*, vol. 5, p. 585, Nov. 2021.
[6] J. Edmonds, "Paths, trees, and flowers," *Canadian J. of Mathematics*, vol. 17, p. 449–467, 1965.
[7] D. Poulin and Y. Chung, "On the iterative decoding of sparse quantum codes," *QIC*, vol. 8, no. 10, p. 987, 2008.
[8] K.-Y. Kuo and C.-Y. Lai, "Refined belief propagation decoding of sparse-graph quantum codes," *IEEE J. Selected Areas in Inf. Theory*, vol. 1, no. 2, pp. 487–498, 2020.
[9] ——, "Exploiting degeneracy in belief propagation decoding of quantum codes," *npj Quantum Inf.*, vol. 8, no. 1, Sep. 2022.
[10] T.-H. Huang, T.-A. Hu, and Y.-L. Ueng, "Branch-assisted sign-flipping belief propagation decoding for topological quantum codes based on hypergraph product structure," *IEEE Trans. on Quantum Engineering*, vol. 4, pp. 1–15, 2023.
[11] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, "Fifteen years of quantum LDPC coding and improved decoding strategies," *IEEE Access*, vol. 3, pp. 2492–2519, 2015.
[12] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Phys. Rev. A*, vol. 54, pp. 1098–1105, Aug. 1996.
[13] N. Raveendran and B. Vasić, "Trapping sets of quantum LDPC codes," *Quantum*, vol. 5, p. 562, Oct. 2021.
[14] A. A. Kovalev and L. P. Pryadko, "Quantum Kronecker sum-product low-density parity-check codes with finite rate," *Phys. Rev. A*, vol. 88, p. 012311, Jul. 2013.
[15] C.-F. Kung, K.-Y. Kuo, and C.-Y. Lai, "On belief propagation decoding of quantum codes with quaternary reliability statistics," in *2023 12th Intl. Symp. on Topics in Coding (ISTC)*, Sep. 2023.