

Efficient Algorithms for the Bee-Identification Problem

Han Mao Kiah*, Alexander Vardy[†], and Hanwen Yao[†]

*School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore

[†]Department of Electrical & Computer Engineering, University of California San Diego, LA Jolla, CA, USA

Emails: hmkih@ntu.edu.sg, avardy@ucsd.edu, hay125@eng.ucsd.edu

Abstract

The bee-identification problem, formally defined by Tandon, Tan and Varshney (2019), requires the receiver to identify “bees” using a set of unordered noisy measurements. In this previous work, Tandon, Tan and Varshney studied error exponents and showed that decoding the measurements *jointly* results in a significantly smaller error exponent.

In this work, we study algorithms related to this joint decoder. First, we demonstrate how to perform joint decoding efficiently. By reducing to the problem of finding perfect matching and minimum-cost matchings, we obtain joint decoders that run in time quadratic and cubic in the number of “bees” for the binary erasure (BEC) and binary symmetric channels (BSC), respectively. Next, by studying the matching algorithms in the context of channel coding, we further reduce the running times by using classical tools like peeling decoders and list-decoders. In particular, we show that our identifier algorithms when used with Reed-Muller codes terminates in almost linear and quadratic time for BEC and BSC, respectively.

Finally, for explicit codebooks, we study when these joint decoders fail to identify the “bees” correctly. Specifically, we provide practical methods of estimating the probability of erroneous identification for given codebooks.

I. INTRODUCTION

Imagine M bees, each tagged with a unique barcode, flying in a beehive. We take a picture of the bees and obtain an unordered set of noisy barcodes. The *bee-identification problem* – proposed and formally defined by Tandon *et al.* – requires one to uniquely identify each bee from the noisy measurements [2]. Besides problems involving multiple target tracking [3], [4], the bee-identification problem is also relevant to other applications (see [2], [5] for other examples). One recent possible application is that of pooled testing for viral RNA like COVID-19. In a recent experiment [6], Schmid-Burgk *et al.* developed a procedure where multiple DNA samples are pooled, sequenced and analyzed *en masse* for the COVID-19 infection. In their procedure, barcodes with high Levenshtein distance were inserted in the DNA samples and by decoding the barcodes individually, they were able to identify the viral DNA samples. Later, the procedures were validated by other groups who performed similar experiments [7]–[10].

Indeed, to recover the original barcodes, a naive approach is to look at each barcode separately and decode them *independently*. However, certain bees/DNA samples may be assigned to the same barcode

Parts of this work were presented in the IEEE International Symposium on Information Theory (ISIT2021) [1].

and in this case, we fail to identify all the bees/DNA samples. In contrast, one can look at all the barcodes *jointly* and determine the best way to assign the barcodes so that likelihood of correct identification is maximized. The latter is termed as joint decoding and in [2], Tandon *et al.* showed that joint decoding results in significantly smaller probability of wrong or failed identification. Specifically, they quantified the gap between the error exponents of independent and joint decoding. Interestingly, in a follow up work, Tandon *et al.* showed that the error exponents are the same for both independent and joint decoding when bees are absent with certain probability [11].

In [2], Tandon *et al.* wrote that the lower error exponent of joint decoding comes at a “cost of increased computational complexity”. They then posited that joint decoding entails a computationally prohibitive exhaustive search amongst the $M!$ possible permutations and explored ideas that combine both independent and joint decoding.

Fortunately, an exhaustive search is not necessary and in this work, we demonstrate that *efficient joint decoding is achievable*. Specifically, for the binary erasure and binary symmetric channels, we reduce the bee-identification problem to the problem of finding a perfect matching and minimum-cost matching, respectively. Hence, applying the well-known Hopcraft-Karp algorithm [13] and Hungarian method [14], respectively, we can identify the bees in time polynomial in M .

We then study the (minimum-cost) matching problem in the context of channel coding and show that the complexity of bee-identification problem can be further reduced. In particular, for the binary erasure channel, we showed that when we deploy the celebrated Reed-Muller codes, the bee-identification problem can be resolved in *almost $O(M)$ time on average*. This is essentially optimal as $\Omega(M)$ time is required to read all M barcodes. Therefore, not only is the “cost of increased computational complexity” for joint decoding acceptable, but, in some cases, the additional complexity cost is negligible.

Finally, we also investigate the probability of erroneous identification for these joint decoders. Specifically, in this work, by relating these probability computations to the problem of permanent computation, we develop practical methods of analyzing and estimating these error probabilities for any *code of interest*. In contrast, in [11], Tandon *et al.* fixed a certain code rate $R > 0$ and determined a corresponding probability estimate $p(R)$. Then using random coding techniques, they showed the existence of a code whose rate is approximately R with the property: under joint decoding, the probability of erroneous identification is at most $p(R)$.

In the next section, we formally describe the bee-identification problem given in [11] and then state our technical contributions. For the ease of exposition, we study the bee-identification problem for the binary erasure channel (BEC) and binary symmetric channel (BSC). The methods in this paper can be extended for larger alphabets and to perform joint maximum-likelihood decoding for other channels.

Finally, to conclude this introduction, we mention certain work that followed the conference version of this work. In [12], motivated by applications that involve DNA strands, Chrisnata *et al.* studied a version of the bee-identification problem where multiple outputs (from a single input) are available. In the same paper, Chrisnata *et al.* also studied the bee-identification problem in the context of deletion channels.

II. PROBLEM DEFINITION

For an integer M , we let $[M]$ denote the set of integers $\{1, 2, \dots, M\}$. The set of all permutations over $[M]$ is denoted by \mathbb{S}_M and we write a permutation $\sigma \in \mathbb{S}_M$ as $\sigma(1)\sigma(2) \cdots \sigma(M)$.

Consider a binary code $\mathcal{C} \subseteq \{0, 1\}^n$ of length n with M codewords $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M$. Consider, in addition, a binary channel where each output \mathbf{y} given an input \mathbf{x} is received with probability $P(\mathbf{y}|\mathbf{x})$. We send *all* M codewords over the channel and obtain an unordered set of M outputs $\{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_M\}$. Note that \mathbf{y}_i is not necessarily the channel output of \mathbf{x}_i and in fact, the task of the *bee-identification problem* is to find a length- M permutation σ so that $\mathbf{y}_{\sigma(i)}$ is indeed the channel output of the input \mathbf{x}_i for all $i \in [M]$. Assuming the channels are independent, the *joint decoder* finds a length- M permutation σ^* that maximizes the probability $\prod_{i \in [M]} P(\mathbf{y}_{\sigma(i)}|\mathbf{x}_i)$. In other words, the joint decoder returns a permutation σ^* such that

$$\sigma^* \in \arg \max_{\sigma \in \mathbb{S}_M} \prod_{i \in [M]} P(\mathbf{y}_{\sigma^*(i)}|\mathbf{x}_i).$$

In this paper, we first study efficient ways of performing joint decoding, that is, computing the permutation σ^* . Since the input to our problem is a set of M n -bit codewords, a trivial lower bound on complexity is $\Omega(Mn)$ and our running time analysis in most parts will be with respect to the parameter M . Also, as the code size M represents “the number of bees”, we assume a reasonable growth rate of M with respect to n , that is, polynomial in n . Hence, in most parts of the paper, we suppress factors involving n in the big-O notation. We note that this is somewhat different from the setting in [2] where $M = 2^{Rn}$ for some positive rate R .

Next, for a fixed code \mathcal{C} , we also investigate when the joint decoder fails to return the correct permutation σ . Namely, if σ^* is the permutation returned by the joint decoder, we provide estimates on the quantity, $P_{\text{error}}(\mathcal{C})$, the probability of the event where $\sigma \neq \sigma^*$. Similar estimates were given in [11]. Specifically, for a fixed value $0 < R < 1$, Tandon *et al.* found an exponent $e(R) > 0$ such that the following holds: there exists a family of codes $\{\mathcal{C}_n\}_{n \geq 1}$ with blocklengths n and rates approaching R so that $P_{\text{error}}(\mathcal{C}_n) \leq 2^{-e(R)n}$. As their derivations rely on random coding techniques, it is unclear whether their estimates apply to explicit codes. In contrast, we fix a specific code \mathcal{C} in this work and provide practical methods of estimating $P_{\text{error}}(\mathcal{C})$.

A. Our Contributions

We summarize our contributions here.

- For the BEC, we provide a joint decoder – Joint Erasure Decoding Identifier (JEDI) – that runs in $O(M^2)$ time. For the family of r -th order Reed-Muller codes and any small ϵ , we show that on average, JEDI terminates in $O(M^{1+\epsilon})$ time when $r \geq 2$ and in $O(M^{2+\epsilon})$ time when $r = 1$.
- For the BSC, we provide a joint decoder – Joint Minimum-Distance Decoding Identifier (JMIDI) – that runs in $O(M^3)$ time. To improve the running time, we approximate the exact solution using ideas from list-decoding and propose the Joint List Decoding Identifier (JLDI). For the family of r -th order Reed-Muller codes, we show that for sufficiently small crossover probability p , JLDI terminates in $O(M^{2+\epsilon})$ time for any small ϵ and is almost as good as JMIDI (see Theorem 8 for the formal statement).

- Finally, for a fixed code \mathcal{C} , we provide probability estimates on when our joint decoders are erroneous. Specifically, using trellis-based techniques, we provide methods to compute upper and lower bounds for error probability in $O(M2^M)$ and $O(M^{3/2}2^{2M})$ time, respectively. We also derive a closed formula that computes a weaker upper bound efficiently.

III. JOINT ERASURE DECODING IDENTIFIER

In this section, we consider the binary erasure channel (BEC). Even though the case for BECs was not studied in [2], we investigate the joint decoder for the erasure channel as it illustrates certain key graph theoretic concepts for the Joint Minimum-Distance Decoding Identifier described in Section IV.

Given an integer M , a *balanced bipartite graph* \mathcal{G} of order M is an undirected graph with $2M$ nodes: M left and M right nodes, where every edge connects a left node to a right node. A *matching* \mathcal{M} of \mathcal{G} is a subset of edges where no two edges are incident on the same node. Clearly, any matching of a balanced bipartite graph \mathcal{G} of order M has at most M edges. If a matching \mathcal{M} contains exactly M edges, we say that \mathcal{M} is *perfect*.

Let us label the left and right nodes of a balanced bipartite graph \mathcal{G} of order M with the M inputs x_1, x_2, \dots, x_M and the M outputs y_1, y_2, \dots, y_M , respectively. Suppose that we have a perfect matching \mathcal{M} of \mathcal{G} . Then we can write the edges of \mathcal{M} as $(x_1, y_{\sigma(1)}), (x_2, y_{\sigma(2)}), \dots, (x_M, y_{\sigma(M)})$ and it follows from the definition of a matching that σ is a permutation of length M . In other words, we can represent a perfect matching with a length- M permutation. Conversely, given a length- M permutation σ , we obtain a perfect matching of \mathcal{G} if $(x_i, y_{\sigma(i)})$ is an edge of \mathcal{G} for all $i \in [M]$. Therefore, for the rest of this paper, we use permutations and matchings interchangeably.

We are now ready to describe the main contribution of this section: an efficient implementation of a joint decoder for erasures.

Joint Erasure Decoding Identifier (JEDI).

INPUT: A codebook $\mathcal{C} = \{x_1, x_2, \dots, x_M\} \subseteq \{0, 1\}^n$ of size M and a set of M channel outputs $\{y_1, y_2, \dots, y_M\} \subseteq \{0, 1, ?\}^n$.

OUTPUT: A permutation σ such that y_i matches $x_{\sigma(i)}$ for all $i \in [M]$ if there is a unique σ . Otherwise, the decoder declares FAILURE.

- (1) We draw a balanced bipartite graph \mathcal{G} of order M . Here, the M codewords are the left nodes while the M channel outputs are the right nodes. For $i, j \in [M]$, we draw an edge between x_i and y_j if and only if y_j matches x_i . Here, we say that y matches x if both y coincides with x on positions that are not erased. Henceforth, we refer to this graph \mathcal{G} as the *input-output graph*.
- (2) Determine if there is a unique perfect matching in \mathcal{G} . If the matching σ is unique, return σ . If the matching is not unique, return FAILURE.

Here, we discuss the running time of JEDI. For general codebooks, Step 1 can be implemented in $O(M^2)$ time. Next, we let \mathcal{G} be the input-output graph constructed in Step 1 and E to be the number of edges in \mathcal{G} . Before we analyze Step 2, we first state some properties of \mathcal{G} . For each codeword $x \in \mathcal{C}$, let $Y(x)$ be its corresponding channel output and we have that $Y(x)$ matches x . Therefore, the set of edges $\{(x, Y(x)) : x \in \mathcal{C}\}$ is a perfect matching of \mathcal{G} . Hence, we have two sub-tasks in Step 2: finding a perfect matching (since it exists) and determining if the matching is unique. For the first sub-task, we can use

the Hopcraft-Karp algorithm [13] to find a perfect matching in $O(E\sqrt{M})$ time. For the second sub-task, we can follow the methods described in Fukada [15] and Hoang *et al.* [16], and then determine if another perfect matching exists in $O(M + E) = O(E)$ time (since $M \leq E$). Hence, combining the analysis of both sub-tasks, we have that Step 2 can be implemented in $O(E\sqrt{M}) = O(M^{2.5})$ time. Therefore, this simple analysis shows that JEDI runs in $O(M^{2.5})$ time.

Nevertheless, the complexity of JEDI can be further reduced. We do so by *improving the running time of Step 2*. Crucially, we exploit the fact that \mathcal{G} contains a perfect matching. Now, if we are able to determine early if there is more than one perfect matching, we need not continue to find a perfect matching. To do so, we modify the classic peeling decoders used in graph-based codes [17]. Intuitively, we search for degree-one nodes in the graph \mathcal{G} . For any such node u , the edge uv incident to u necessarily belongs to a perfect matching and hence, we add it to the matching. We then remove both nodes u and v , and all other edges incident to v and repeat the search for degree-one nodes. We have two scenarios. In the first scenario, we remove all nodes from \mathcal{G} and end up with a perfect matching. In the second scenario, all remaining nodes have degree at least two and it can be shown that \mathcal{G} contains at least two perfect matchings (see Section III-B). Thus, in this case, we can terminate our search earlier. A formal description is given below.

Peeling Matching Algorithm (PMA).

INPUT: A bipartite graph \mathcal{G} (with M left and M right vertices) that contains at least one perfect matching.

OUTPUT: A perfect matching \mathcal{M} of \mathcal{G} if it is unique. Otherwise, FAILURE is declared.

- (1) Initialize \mathcal{M} to the empty set.
- (2) Find a node u in \mathcal{G} with degree one. Here, u may be a left *or* right node. If there is no such node, go to Step 6.
- (3) Let uv be the unique edge incident to u and add uv to \mathcal{M} .
- (4) Remove nodes u and v and all edges incident to v .
- (5) Repeat Step 2.
- (6) If \mathcal{M} is a perfect matching, return \mathcal{M} . Otherwise, $|\mathcal{M}| < M$ and we declare FAILURE.

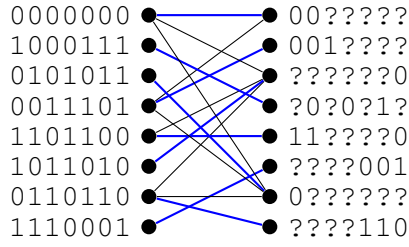
Example 1. Consider the simplex code of length seven. More concretely, we consider the linear code

with $M = 8$ codewords generated by the matrix $\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$.

- (a) Suppose the channel outputs are:

$$\begin{array}{cccc} 00?????, & 001????, & ??????0, & ?0?0?1?, \\ 11????0, & ???001, & 0??????, & ???110. \end{array}$$

Then the bipartite graph \mathcal{G} constructed in Step 1 of JEDI is given below. Highlighted in blue is the unique bipartite matching \mathcal{M} in \mathcal{G} .



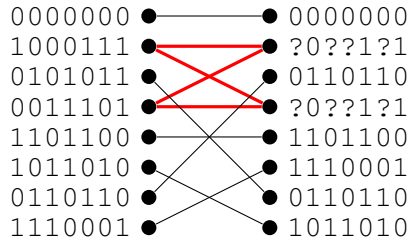
Here, we list the edges *in the order* they are added to \mathcal{M} according to PMA.

$$(1000111, ?0?0?1?), (0101011, 0??????), (1011010, ??????0), (1110001, ?????001), \\ (0011101, 001????), (1101100, 11????0), (0110110, ?????110), (0000000, 00?????).$$

(b) Suppose the channel outputs are:

$$0000000, ?0??1?1, 0110110, ?0??1?1, \\ 1101100, 1110001, 0110110, 1011010.$$

Then the bipartite graph \mathcal{G} constructed is given below.



Highlighted in **red** is the edges remaining after all degree-one nodes are removed. Hence, PMA declares FAILURE.

The following lemma on the correctness of PMA and its running time can be proved using the notion of stopping sets in peeling decoders [17]. For completeness, we provide a detailed proof in Section III-B.

Lemma 1. *Let \mathcal{G} be a balanced bipartite graph of order M with E edges. Suppose that \mathcal{G} contains at least one perfect matching. If the perfect matching is unique, then the output \mathcal{M} of PMA is the perfect matching. Otherwise, PMA declares FAILURE. Furthermore, PMA terminates in $O(E)$ time.*

Therefore, since $E \leq M^2$, we have that JEDI terminates in $O(M^2)$ time. Now, this running time analysis assumes the worst case where \mathcal{G} is a complete bipartite graph. By design, the codebook \mathcal{C} is chosen such that most erasure patterns are correctable with high probability. In other words, each right node or channel output is expected to match with exactly one left node or codeword, and so, we expect the graph \mathcal{G} to be sparse.

It turns out that the expected number of edges in \mathcal{G} is given by the distance enumerator (see for example [18]). Specifically, given a code \mathcal{C} of length n , we define B_i to be the number of pairs of (not necessarily distinct) codewords of distance i . So, we have that $B_0 = M$ and $\sum_{i=0}^n B_i = M^2$. We then define the *distance enumerator of code \mathcal{C}* to be polynomial $B(z) = \sum_{i=0}^n B_i z^i$.

Lemma 2. *Consider a BEC with erasure probability p . If the distance enumerator of code \mathcal{C} is $B(z)$, then expected number of edges in \mathcal{G} constructed in JEDI is given by $B(p)$.*

Proof. Consider two codewords x and z and let \tilde{x} be the channel output of x . We first compute the probability that there is an edge between the nodes z and \tilde{x} in JEDI. Suppose the distance between x and z is d . Then there is an edge between z and \tilde{x} if the d bits where x and z differ are erased. In other words, there is an edge between z and \tilde{x} with probability p^d . By linearity of expectation, we have that the expected number of edges is $\sum_{x,z \in \mathcal{C}} z^{d_H(x,z)} = B(z)$. Here, $d_H(x, z)$ denotes the Hamming distance of x and z . \square

Remark 2. Alternatively, given $\mathcal{C} = \{x_1, x_2, \dots, x_M\}$, we can define an $(M \times M)$ -matrix T_{BEC} whose (i, j) -entry is given by $p^{d_H(x_i, x_j)}$. Then the sum of all M^2 entries in T_{BEC} yields the distance enumerator for \mathcal{C} . While this method of determining the expected number of edges is computationally equivalent, it turns out the permanent of the matrix T_{BEC} can be used to estimate the error probability of JEDI. We describe this in detail in Section V.

Next, we consider a family of block codes and we determine sufficient conditions such the expected running of JEDI is linear in the block sizes M_n .

Theorem 3. *Consider a BEC with erasure probability p . Let \mathcal{C} be a linear code of size M with minimum distance d . Then the expected number of edges of \mathcal{G} is at most $M + \binom{M}{2}p^d$. Hence, the expected running time of JEDI is $O\left(M + \binom{M}{2}p^d + Mn^3\right)$.*

Furthermore, suppose that $\{\mathcal{C}_n\}_{n \geq 0}$ is a family of linear codes such that \mathcal{C}_n is a linear code of size M_n with minimum distance d_n . If $\binom{M_n}{2}p^{d_n} = o(1)$, then the expected running time of JEDI tends to $O(M_n n^3)$.

Proof. First, we consider a code \mathcal{C} with minimum distance d . Then we have that $B_1 = B_2 = \dots = B_{d-1} = 0$. Since $p^i \leq p^d$ for $i \geq d$, we have that $B(p) = B_0 + \sum_{i=d}^n B_i p^i \leq M + \binom{M}{2}p^d$. Therefore, the expected number of edges in \mathcal{G} is at most $M + \binom{M}{2}p^d$. Hence, Step 2 runs in $O\left(M + \binom{M}{2}p^d\right)$ time.

So, it remains to improve the running time of Step 1, i.e. the time to construct the input-output graph \mathcal{G} . Since \mathcal{C} is a linear code of length n , then for any channel output y , we can find all x such that y matches x in $O(n^3)$ time by matrix inversion. Therefore, \mathcal{G} can be constructed in $O(Mn^3)$ time.

The asymptotic analysis for $\{\mathcal{C}_n\}_{n \geq 0}$ follows from the preceding argument. \square

A. Reed-Muller Codes

In this subsection, we apply Theorem 3 to the ubiquitous class of Reed-Muller codes [19] and derive the expected running time of JEDI on this class of linear codes.

Theorem 4. *Fix $r \geq 1$ and consider the family of r -th order Reed-Muller codes. Then for any $\epsilon > 0$, the expected running time of JEDI is*

$$\begin{cases} O(M^{1+\epsilon}) & \text{when } r \geq 2, \\ O(M^{2+\epsilon}) & \text{when } r = 1. \end{cases}$$

Proof. Consider $1 \leq r \leq m$. Recall that the Reed-Muller code $\mathcal{RM}(r, m)$ has the following parameters: $n = 2^m$, $\log M = \sum_{i=0}^r \binom{m}{i}$ and $d = 2^{m-r}$.

First, we demonstrate the expected running time tends to $O(Mn^3)$. Following Theorem 3, it remains to show that $\binom{M}{2}p^d$ tends to zero as $m \rightarrow \infty$, or equivalently,

$$\log \binom{M}{2} + d \log p \rightarrow -\infty. \quad (1)$$

Now, for $\mathcal{RM}(r, m)$, we have that $\log M \leq (r+1)m^r$. Since $d = 2^{m-r}$, the left hand side of (1) is upper bounded by $2(r+1)m^r + (2^{-r} \log p)2^m$. When $p < 1$, this upper bound tends to $-\infty$ and hence, we have that $\binom{M}{2}p^d$ tends to zero, as required. This means that the input-output graph \mathcal{G} has expected number of edges at most M .

Next, we bound the time required to construct \mathcal{G} . Let ϵ be a positive constant.

- When $r \geq 2$, we use matrix inversion to construct \mathcal{G} as in the proof of Theorem 3 and we claim that $n^3 = O(M^\epsilon)$. Indeed, $M^\epsilon = 2^{\epsilon(\sum_{i=0}^r \binom{m}{i})} \geq 2^{\gamma m^r}$ for some constant γ . On the other hand, $n^3 = 2^{3m}$. Since $3m = o(\gamma m^r)$, we have that $2^{3m} = O(2^{\gamma m^r}) = O(M^\epsilon)$.
- When $r = 1$, it is no longer true that $n^3 = O(M^\epsilon)$. Instead of using matrix inversion to construct \mathcal{G} , we use the Fast Hadamard Transform (FHT) [20], [21] to determine the edges. Specifically, for each channel output \mathbf{y} , we can use FHT to find all \mathbf{x} that matches \mathbf{y} in $O(M \log M)$ time. Therefore, \mathcal{G} can be constructed in $O(M^2 \log M) = O(M^{2+\epsilon})$ time and we have the expected running time of JEDI as desired. \square

To end this subsection, we comment that the derived running time is essentially optimal. As mentioned earlier, since we have to read all M codewords of length n , a lower bound for the running time of any joint decoder is trivially $\Omega(Mn)$. For Reed-Muller codes of order $r \geq 2$, we have $n = o(M)$ and the expected running time of $O(M^{1+\epsilon})$ is almost optimal. When $r = 1$, we note that $n = \Theta(M)$ and thus, $\Omega(Mn) = \Omega(M^2)$. Again, the expected running time of $O(M^{2+\epsilon})$ is almost optimal.

B. Correctness and Running Time of PMA

For completeness, we provide a detailed proof of Lemma 1. Following Fukada [15] and Hoang *et al.* [16], we define the notion of alternating cycle. Formally, consider a bipartite graph \mathcal{G} with a perfect matching \mathcal{M} . A cycle \mathcal{O} in \mathcal{G} is *alternating* with respect to \mathcal{M} if the edges in \mathcal{O} alternate between in \mathcal{M} and not in \mathcal{M} . We have the following lemma.

Lemma 5. *Let \mathcal{M} be a perfect matching in a balanced bipartite \mathcal{G} . Then \mathcal{M} is the unique perfect matching in \mathcal{G} if and only if there is no alternating cycle with respect to \mathcal{M} .*

Correctness of PMA. Applying Lemma 5, it suffices to show the following claim.

Let \mathcal{G} be a balanced bipartite graph with a perfect matching \mathcal{M} . If all the degrees of \mathcal{G} are at least two, then there is an alternating cycle with respect to \mathcal{M} .

Indeed, we construct an alternating cycle as follow. Pick any left node u_1 in \mathcal{G} . Since \mathcal{M} is a perfect matching, we can find a right node v_1 such that u_1v_1 belongs to \mathcal{M} . Now, as the degree of v_1 is at least two, we can find u_2 such that u_2v_1 is an edge not belonging to \mathcal{M} . We then repeat the process to find v_2 and u_3 such that $u_2v_2 \in \mathcal{M}$ and $u_3v_2 \notin \mathcal{M}$. Since the degrees of all nodes are at least two, we are always able to find a left node u_i and eventually, we have two left nodes that coincide and obtain an alternating cycle with respect to \mathcal{M} .

Running Time of PMA. We briefly describe a data structure that implements PMA in time linear in the number of edges. Recall that \mathcal{G} has E edges and $2M$ nodes with $E \geq V$.

We maintain an *adjacency list* for the nodes. In other words, for each node v , we maintain a list $N(v)$ of nodes adjacent to v . Also, we maintain a queue Q of degree-one nodes.

Whenever Q is nonempty, we remove the first node u and its neighbor v and update the adjacency lists of the neighbors of v . If any node becomes degree-one, we add it to the Q . We continue this process until the queue is empty. Since the number of updates to the adjacency lists is at most the number of edges, the running time of PMA is $O(M + E) = O(E)$.

IV. JOINT MINIMUM-DISTANCE DECODING IDENTIFIER

We propose an efficient joint decoder for the binary symmetric channel (BSC). While our exposition assumes a BSC channel, we remark that the decoder can be modified to serve as a joint maximum likelihood decoder for other channels (see Section VI).

As with Section III, we reduce the problem of permutation recovery to that of finding a minimum-cost matching. Specifically, consider a balanced bipartite graph \mathcal{G} of order M . In addition, we associate each edge e in \mathcal{G} with a cost $c(e)$ and the cost of a matching \mathcal{M} is the sum of the costs of all edges in \mathcal{M} . Suppose that \mathcal{G} contains at least one perfect matching. A perfect matching in \mathcal{G} is *minimum-cost* if its cost is at most the cost of any other perfect matching in \mathcal{G} . When \mathcal{G} is a complete bipartite graph, the problem of finding a minimum-cost matching in \mathcal{G} is also known as the *assignment* problem and the Hungarian method finds a minimum-cost assignment in $O(M^3)$ time [14].

As with before, we use the codewords and channel outputs as the left and right nodes of a balanced bipartite graph \mathcal{G} . Then for any codeword-output pair (\mathbf{x}, \mathbf{y}) , we connect them with an edge of cost $d_H(\mathbf{x}, \mathbf{y})$. Then the problem of finding a permutation σ that maximizes the probability $\prod_{i \in [M]} P(\mathbf{y}_{\sigma(i)} | \mathbf{x}_i)$ is equivalent to minimizing the cost of a perfect matching in \mathcal{G} . A formal description of the decoder is given below.

Joint Minimum-Distance Decoding Identifier (JMEDI).

INPUT: A codebook $\mathcal{C} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M\}$ of size M and a set of M channel outputs $\{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_M\} \subseteq \{0, 1\}^n$.

OUTPUT: A permutation σ such that the quantity $\sum_{i \in [M]} d_H(\mathbf{x}_i, \mathbf{y}_{\sigma(i)})$ is minimized.

- (1) We draw a balanced bipartite graph \mathcal{G} of order M . Here, the M codewords are the left nodes while the M channel outputs are the right nodes. For $i, j \in [M]$, we draw an edge between \mathbf{x}_i and \mathbf{y}_j and set its cost to be $d_H(\mathbf{x}_i, \mathbf{y}_j)$. Again, we refer to this bipartite graph \mathcal{G} as the *input-output graph*.
- (2) Find a minimum-cost matching in \mathcal{G} .

We discuss the running time of JMEDI. In Step 1, we can compute the distance between all pairs of words in $O(M^2n)$ time. In Step 2, we can apply the Hungarian method [14] and hence, we have the following theorem.

Theorem 6. *JMEDI finds a permutation in $O(M^3 + M^2n)$ time.*

Unlike the input-output graph constructed for the BEC, the input-output graph obtained by the JMEDI is necessarily complete. So, to improve the running time for the case of BSC channels, we relax our goal

of finding the exact solution. Instead, we approximate it by determining a minimum-cost matching in a sparse subgraph \mathcal{H} of \mathcal{G} .

Specifically, for this sparse subgraph \mathcal{H} , we consider only edges whose cost is at most R for some $R < n$. Then the degree of each right node/channel output \mathbf{y} is given by the number of codewords whose distance is at most R from \mathbf{y} . When \mathcal{C} is a code with minimum distance d and $R \leq (d-1)/2$, this number is one. When $R > (d-1)/2$, this quantity is studied in the context of *list decoding*.

Formally, a \mathcal{C} is (R, L) -list-decodable if for all $\mathbf{y} \in \{0, 1\}^n$, we have that $|\{\mathbf{x} \in \mathcal{C} : d_H(\mathbf{x}, \mathbf{y}) \leq R\}|$ is at most L . Then we modify Step 1 of JMDI by only including edges with weight at most R . Let \mathcal{H} be the resulting bipartite graph and from the list-decoding property of \mathcal{C} , we have that \mathcal{H} has at most ML edges. We then proceed as in Step 2 to find a minimum-cost matching and we call this method *joint list decoding identifier* (JLDI). For sparse bipartite graphs with V nodes and E edges, a minimum-cost matching can be found in $O(V^2 \log V + VE)$ time [22], [23] and hence, JLDI terminates in $O(M^2(\log M + L + n))$ time.

Example 3. Consider the linear code \mathcal{C} with $M = 4$ codewords generated by the matrix $\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$.

Suppose the channel outputs are:

$$\mathbf{y}_1 = 10000, \quad \mathbf{y}_2 = 11101, \quad \mathbf{y}_3 = 00011, \quad \mathbf{y}_4 = 10001.$$

Below we present the bipartite graph \mathcal{G} constructed by JMDI. To reduce clutter, we use a 4×4 -table whose (i, j) entry is given by the cost of the edge $(\mathbf{x}_i, \mathbf{y}_j)$, i.e. $d_H(\mathbf{x}_i, \mathbf{y}_j)$. We refer to this table as the cost matrix of \mathcal{G} .

	\mathbf{y}_1	\mathbf{y}_2	\mathbf{y}_3	\mathbf{y}_4
$\mathbf{x}_1 = 00000$	1	4	2	2
Cost Matrix of \mathcal{G} : $\mathbf{x}_2 = 11100$	2	1	5	3
$\mathbf{x}_3 = 00111$	4	3	1	3
$\mathbf{x}_4 = 11011$	3	2	2	2

Highlighted in **blue** are the edges in a minimum-cost matching of \mathcal{G} . Here, the minimum-cost matching \mathcal{M} is given by $\{(\mathbf{x}_i, \mathbf{y}_i) : i \in [4]\}$.

Now, we can verify that \mathcal{C} is a $(2, 3)$ -list-decodable code. Hence, if we apply JLDI with radius $R = 2$, we obtain the bipartite graph \mathcal{H} whose cost-matrix is as follows.

	\mathbf{y}_1	\mathbf{y}_2	\mathbf{y}_3	\mathbf{y}_4
$\mathbf{x}_1 = 00000$	1	-	2	2
Cost Matrix of \mathcal{H} : $\mathbf{x}_2 = 11100$	2	1	-	-
$\mathbf{x}_3 = 00111$	-	-	1	-
$\mathbf{x}_4 = 11011$	-	2	2	2

Indeed, we observe that the degree of \mathbf{y}_i , $i \in [4]$, is at most three, corroborating the list-decoding property of \mathcal{C} . In this case, we have that the minimum-cost matching of \mathcal{H} is also \mathcal{M} .

However, a minimum-cost matching in \mathcal{H} may not be a minimum-cost matching in \mathcal{G} . In other words, JLDI may not return the same output as JMDI. Nevertheless, such cases occur with small probability and we provide upper bounds on the probabilities of such events.

Theorem 7. Consider a BSC channel with crossover probability p . Let \mathcal{C} be an (R, L) -list-decodable code of length n , size M and $R > pn$. Set $\gamma = R/(pn) - 1$. Then JLDI terminates in $O(M^2(\log M + L + n))$ time. Furthermore,

$$\text{Prob}(\text{JLDI correctly finds } \sigma \mid \text{JMDI correctly finds } \sigma) \geq (1 - \exp(-\gamma^2 pn/3))^M. \quad (2)$$

Proof. The running time analysis of JLDI is described in the preceding paragraphs.

To derive the probability estimates, we consider the random variable Z_i that measures the number of errors in the output of codeword \mathbf{x}_i , $i \in [M]$. In other words, $Z_i = d_H(\mathbf{x}_i, \mathbf{y}_{\sigma(i)})$. Let \mathcal{G} and \mathcal{H} be the bipartite graphs constructed in JMDI and JLDI, respectively. To simplify our arguments, we assume that the minimum-cost matchings in both graphs are unique and let them be $\mathcal{M}_{\mathcal{G}}$ and $\mathcal{M}_{\mathcal{H}}$.

First, we argue that if $Z_i \leq R$ for all $i \in [M]$ and JMDI is correct, then JLDI is necessarily correct. Since JMDI is correct, we have that the matching $\mathcal{M}_{\mathcal{G}}$ corresponds to σ . Also, for all i , since $Z_i \leq R$, we have the edge $(\mathbf{x}_i, \mathbf{y}_{\sigma(i)})$ has cost at most R . Therefore, the matching $\mathcal{M}_{\mathcal{G}}$ is still present in the graph \mathcal{H} and so, the minimum-cost matching $\mathcal{M}_{\mathcal{H}}$ is identical to $\mathcal{M}_{\mathcal{G}}$ and corresponds to σ . Hence, JLDI is correct.

Next, we lower bound the probability that all values of Z_i is at most R . Fix $i \in [M]$. Using Chernoff's bound (see for example, [24]), we have that $P(Z_i \geq R) = P(Z_i \geq (1 + \gamma)pn) \leq \exp(-\gamma^2 pn/3)$ and so, $P(Z_i \leq R) \geq 1 - \exp(-\gamma^2 pn/3)$. Since the values of Z_i are independent of each other, the lower bound (2) follows. \square

A. Reed-Muller Codes

Similar to before, we verify that the class of Reed-Muller codes satisfy the conditions of Theorem 7.

Theorem 8. Fix $r \geq 1$ and consider the family of r -th order Reed-Muller codes. Consider further a BSC with crossover probability $p < 1/2^r$.

For any $\epsilon > 0$, JLDI runs in $O(M^{2+\epsilon})$ time and event (2) occurs with probability approaching one.

To demonstrate the result, we apply the following result on the list-decoding capabilities of Reed-Muller codes.

Theorem 9 (Bhomick and Lovett [25]). Fix r and α and set $R = (1/2^r - \alpha)n$. Then there is a constant $L_{r,\alpha}$ (dependent only on r and α) such that the Reed-Muller code $\mathcal{RM}(r, m)$ is $(R, L_{r,\alpha})$ -list-decodable for all m .

Proof of Theorem 8. Choose some small α so that $p < 1/2^r - \alpha$ and set $R = (1/2^r - \alpha)n$. Then Theorem 9 states that the list size is upper bounded by a constant independent of n and M . Hence, applying Theorem 7, we have the running time is $O(M^2(\log M + L + n)) = O(M^2(\log M + n))$.

When $r \geq 2$, we have that $n = O(M^\epsilon)$ as in the proof of Theorem 4 and hence, the running time is $O(M^{2+\epsilon})$. When $r = 1$, as before, we use FHT to compute the costs of the edges. Specifically, for each channel output \mathbf{y} , we use FHT to compute $d_H(\mathbf{x}, \mathbf{y})$ for all codewords \mathbf{x} in $O(M \log M)$ time. Therefore, \mathcal{G} can be constructed in $O(M^2 \log M) = O(M^{2+\epsilon})$ time.

Next, Theorem 7 also states that event (2) occurs with probability at least $\theta_n \triangleq (1 - \exp(-\gamma^2 pn/3))^M$ for some constant γ . Recall that M is the code size $2^{\sum_{i=0}^r \binom{m}{i}} \leq 2^{(r+1)m^r} = O(2^{\lambda n})$ for all any constant

λ . So, we can choose λ small enough so that $2^\lambda < \exp(\gamma^2 p/3)$. Therefore, $M \exp(-\gamma^2 pn/3)$ approaches zero and we have

$$\begin{aligned} \lim_{n \rightarrow \infty} \theta_n &= \lim_{n \rightarrow \infty} (1 - \exp(-\gamma^2 pn/3))^M \\ &= \lim_{n \rightarrow \infty} \left(\left(1 - \frac{1}{\exp(\gamma^2 pn/3)} \right)^{\exp(\gamma^2 pn/3)} \right)^{\frac{M}{\exp(\gamma^2 pn/3)}} \\ &= \lim_{n \rightarrow \infty} e^{-M \exp(-\gamma^2 pn/3)} = 1, \text{ as desired} \quad \square \end{aligned}$$

V. PROBABILITY OF ERRONEOUS IDENTIFICATION

In this section, we provide probability estimates for the event where the joint decoders (described in Sections III and IV) fail to identify the codewords / bees. Specifically, throughout this section, we let \mathcal{C} denote a code with M length- n codewords $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M$. As before, we send these M codewords through a noisy channel and let $\{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_M\}$ be the corresponding set of outputs. Then there exists a permutation $\sigma \in \mathbb{S}_M$ such that $\mathbf{y}_{\sigma(i)}$ is the noisy output of \mathbf{x}_i for $i \in [M]$.

Let σ^* be the permutation / matching returned by the joint decoder and our task to estimate the probability that $\sigma^* \neq \sigma$, or, equivalently, the probability that $\sigma^* \sigma^{-1} \neq \text{id}$. Here, σ^{-1} denotes the inverse permutation of σ while id denotes the identity permutation. Without loss of generality, we assume that $\sigma = \text{id}$ and we consider the following error events.

- *Binary erasure channels (BEC)*. Recall that JEDI returns the correct perfect matching / permutation id if and only if id is the unique perfect matching in the input-output graph \mathcal{G} constructed in Step 1. In other words, if the graph contains another matching $\sigma \in \mathbb{S}_M^* \triangleq \mathbb{S}_M \setminus \{\text{id}\}$, JEDI declares failure. Hence, we are interested in the event where σ appears as a matching in the graph \mathcal{G} and we denote this event by $I(\sigma)$. Note that $\text{Prob}(I(\text{id})) = 1$.
- *Binary symmetric channels (BSC)*. Recall that JMDI returns a minimum-cost perfect matching from the weighted input-output graph constructed in Step 1. Hence, if the cost of the identity matching id is smaller than all other matchings, JMDI necessarily returns id . Hence, for some non-identity permutation $\sigma \in \mathbb{S}_M^*$, we study the event where the cost of matching σ is at most the cost of the identity matching and for convenience, we also denote this event by $I(\sigma)$. As with the binary erasure channel, we have that $\text{Prob}(I(\text{id})) = 1$.

Therefore, our task is to estimate the quantity $P_{\text{error}}(\mathcal{C}) \triangleq \text{Prob}\left(\bigcup_{\sigma \in \mathbb{S}_M^*} I(\sigma)\right)$. Now, by the union bound, we have that $P_{\text{error}}(\mathcal{C}) \leq \sum_{\sigma \in \mathbb{S}_M^*} \text{Prob}(I(\sigma))$. If we further define $U \triangleq \sum_{\sigma \in \mathbb{S}_M} \text{Prob}(I(\sigma))$, it is straightforward to see that

$$P_{\text{error}}(\mathcal{C}) \leq U - 1. \quad (3)$$

On the other hand, by Bonferroni inequalities / principles of inclusion-exclusion, we have that

$$P_{\text{error}}(\mathcal{C}) \geq \sum_{\sigma \in \mathbb{S}_M^*} \text{Prob}(I(\sigma)) - \sum_{\substack{\sigma \neq \tau, \\ \sigma, \tau \in \mathbb{S}_M^*}} \text{Prob}(I(\sigma) \wedge I(\tau)).$$

Proceeding similar as before, we define $V \triangleq \sum_{\sigma, \tau \in \mathbb{S}_M} \text{Prob}(I(\sigma) \wedge I(\tau))$ and we have

$$\begin{aligned}
V &= \sum_{\substack{\sigma \neq \tau, \\ \sigma, \tau \in \mathbb{S}_M^*}} \text{Prob}(I(\sigma) \wedge I(\tau)) + 2 \sum_{\sigma \in \mathbb{S}_M} \text{Prob}(I(\sigma) \wedge I(\text{id})) + \sum_{\sigma \in \mathbb{S}_M} \text{Prob}(I(\sigma) \wedge I(\sigma)) - 2\text{Prob}(I(\text{id}) \wedge I(\text{id})) \\
&= \sum_{\substack{\sigma \neq \tau, \\ \sigma, \tau \in \mathbb{S}_M^*}} \text{Prob}(I(\sigma) \wedge I(\tau)) + 2 \sum_{\sigma \in \mathbb{S}_M} \text{Prob}(I(\sigma)) + \sum_{\sigma \in \mathbb{S}_M} \text{Prob}(I(\sigma)) - 2 \\
&= \sum_{\substack{\sigma \neq \tau, \\ \sigma, \tau \in \mathbb{S}_M^*}} \text{Prob}(I(\sigma) \wedge I(\tau)) + 3U - 2.
\end{aligned}$$

Therefore, together with the fact that $\sum_{\sigma \in \mathbb{S}_M^*} \text{Prob}(I(\sigma)) = U - 1$, we have that

$$P_{\text{error}}(\mathcal{C}) \geq (U - 1) - (V - 3U + 2) = 4U - V - 3. \quad (4)$$

Hence, following (3) and (4), to provide the required estimates on $P_{\text{error}}(\mathcal{C})$, it suffices to determine U and V . However, U and V involve $M!$ and $(M!)^2$ summands, respectively, and in fact, we show later that U can be expressed as a permanent function of a certain $M \times M$ matrix T . Unfortunately, determining the permanent of a general matrix is computationally intractable [26] and the state-of-the-art methods of computing permanents, due to Nijenhuis-Wilf [27] and Glynn [28], have running time $O(M2^{M-1})$. In a recent work [29], we studied methods of computing permanents on trellises. While our trellis-based techniques did not significantly improve the running time of state-of-the-art methods for general matrices, we observed that, for structured matrices and permanent-like functions, we can borrow ideas from trellis theory [30] to significantly reduce the running time. We apply these techniques here to determine V . Specifically, in Subsection V-B, we show that V can be computed in $O(M^{1.5}4^M)$ time.

A. Estimating U

In this subsection, we provide estimates for U using the following notion of permanents.

Definition 4. Let T be an $M \times M$ -matrix whose (i, j) -th entry is T_{ij} . Then the *permanent* of T is given by the value

$$\text{per}(T) \triangleq \sum_{\sigma \in \mathbb{S}_M} \prod_{i \in [M]} T_{i\sigma(i)}. \quad (5)$$

Given a code \mathcal{C} of size M with codewords $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M$. Recall that \mathbf{T}_{BEC} denote an $(M \times M)$ -matrix whose (i, j) -entry is given by $p^{d_H(\mathbf{x}_i, \mathbf{x}_j)}$. On the other hand, we let $\mathbf{T}_{\text{BSC}}^{(u)}$ and $\mathbf{T}_{\text{BSC}}^{(\ell)}$ denote two $(M \times M)$ -matrices. The (i, j) -th entry of $\mathbf{T}_{\text{BSC}}^{(u)}$ is given by $(4p(1-p))^{d_H(\mathbf{x}_i, \mathbf{x}_j)/2}$, while the (i, j) -th entry of $\mathbf{T}_{\text{BSC}}^{(\ell)}$ is given by $(p(1-p))^{d_H(\mathbf{x}_i, \mathbf{x}_j)/2}$. Then the next proposition states that the permanents of these matrices can be used to estimate U .

Proposition 10. Let \mathcal{C} be a code and let $0 < p < 1/2$. Recall that $U \triangleq \sum_{\sigma \in \mathbb{S}_M} \text{Prob}(I(\sigma))$.

- (i) If we transmit the codebook \mathcal{C} through a BEC(p), then $U = \text{per}(\mathbf{T}_{\text{BEC}})$.
- (ii) If we transmit the codebook \mathcal{C} through a BSC(p), then $\text{per}(\mathbf{T}_{\text{BSC}}^{(\ell)}) \leq U \leq \text{per}(\mathbf{T}_{\text{BSC}}^{(u)})$.

Proof. In both cases, following the definition of permanent (9), it suffices to demonstrate a certain relationship between $\text{Prob}(I(\sigma))$ and $\prod_{i \in [M]} T_{i\sigma(i)}$.

- (i) First, we consider the binary erasure channel and fix some permutation σ . Then $I(\sigma)$ is the event that the input-output graph \mathcal{G} contains the perfect matching corresponding to σ . In other words,

for all $i \in [M]$, there is an edge between the input \mathbf{x}_i and the output of $\mathbf{x}_{\sigma(i)}$. As we argued in Lemma 2, this probability is given by $p^{d_H(\mathbf{x}_i, \mathbf{x}_{\sigma(i)})}$ and thus, $\text{Prob}(I(\sigma)) = \prod_{i \in [M]} p^{d_H(\mathbf{x}_i, \mathbf{x}_{\sigma(i)})}$. Finally, the proposition then follows from the definition of T_{BEC} .

- (ii) Next, we consider the binary symmetric channel and again, fix some permutation σ . Then the $I(\sigma)$ is the event where the cost of the matching σ is at most the cost of the identity matching. In other words, we have the (Mn) -length word $\mathbf{x}_{\sigma(1)}\mathbf{x}_{\sigma(2)} \cdots \mathbf{x}_{\sigma(M)}$ is closer to the output $\mathbf{y}_1\mathbf{y}_2 \cdots \mathbf{y}_M$ in terms of Hamming distance than the word $\mathbf{x}_1\mathbf{x}_2 \cdots \mathbf{x}_M$. Let D denote the Hamming distance between $\mathbf{x}_{\sigma(1)}\mathbf{x}_{\sigma(2)} \cdots \mathbf{x}_{\sigma(M)}$ and $\mathbf{x}_1\mathbf{x}_2 \cdots \mathbf{x}_M$. Then the probability of event I_σ is given by $\sum_{j=\lceil D/2 \rceil}^D \binom{D}{j} p^j (1-p)^{D-j}$. Now, Barg and Forney [31, Section III] provided the following estimates for the latter quantity. Specifically, they showed that

$$(p(1-p))^{D/2} \leq \sum_{j=\lceil D/2 \rceil}^D \binom{D}{j} p^j (1-p)^{D-j} \leq (4p(1-p))^{D/2}. \quad (6)$$

Then, using $D = \sum_{i \in [M]} d_H(\mathbf{x}_i, \mathbf{x}_{\sigma(i)})$, we obtain the proposition. \square

Unfortunately, as mentioned earlier, determining the exact value of the permanent of a general matrix is computationally slow. Nevertheless, when the code has a certain minimum distance, we are able to use analytic combinatorics [32] to provide an upper bound for U .

Theorem 11. *Suppose that \mathcal{C} is a code with minimum distance d . Set*

$$\theta = \begin{cases} p^d, & \text{if the channel is BEC}(p), \\ (4p(1-p))^{d/2}, & \text{if the channel is BSC}(p), \end{cases} \quad (7)$$

Then

$$U \leq M! \sum_{i=0}^M \frac{\theta^{M-i} (1-\theta)^i}{i!}. \quad (8)$$

Therefore, $P_{\text{error}}(\mathcal{C}) \leq \left(M! \sum_{i=0}^M \frac{\theta^{M-i} (1-\theta)^i}{i!} \right) - 1$.

Proof. Let \mathbf{T} be either T_{BEC} or $T_{\text{BSC}}^{(u)}$. Then using Proposition 10, it suffices to show that $\text{per}(\mathbf{T})$ is at most $\left(M! \sum_{i=0}^M \theta^{M-i} (1-\theta)^i / i! \right)$.

Now, we consider \mathbf{U} whose entries are such that $\mathbf{U}_{ii} = 1$ and $\mathbf{U}_{ij} = \theta$ if $i \neq j$. Then we observe that $\mathbf{T}_{ij} \leq \mathbf{U}_{ij}$ for all $i, j \in [M]$. Thus, $\text{per}(\mathbf{T})$ is at most $\text{per}(\mathbf{U})$. For each permutation $\sigma \in \mathbb{S}_M$, the product $\prod_{i \in [M]} \mathbf{U}_{i\sigma(i)}$ is given by $\prod_{i \neq \sigma_i} \theta = \theta^{M-f(\sigma)}$, where $\theta = p^d$ and $f(\sigma) = |\{i = \sigma_i : i \in [M]\}|$. Note that $f(\sigma)$ is also known as the number of fixed points of a permutation σ .

Let $F_{j,M}$ be the number of permutations of length M with exactly j fixed points. It turns out the following exponential generating function is known (see [32, Theorem 7.12]).

$$\sum_{j,M} \frac{F_{j,M}}{M!} u^j z^M = \frac{e^{(u-1)z}}{1-z}$$

With this result, let us estimate $\sum_{\sigma \in \mathbb{S}_M} \theta^{M-f(\sigma)}$.

$$\begin{aligned}
\sum_{M \geq 0} \frac{1}{M!} \left(\sum_{\sigma \in \mathbb{S}_M} \theta^{M-f(\sigma)} \right) z^M &= \sum_{M,j} \frac{F_{j,M}}{M!} \theta^{M-j} z^M \\
&= \sum_{M,j} \frac{F_{j,M}}{M!} \theta^{-j} (\theta z)^M \\
&= \frac{e^{(\theta^{-1}-1)\theta z}}{1-\theta z} = \frac{e^{(1-\theta)z}}{1-\theta z}
\end{aligned}$$

Therefore, $\sum_{\sigma \in \mathbb{S}_M} \theta^{M-f(\sigma)} = M! [z^M] \frac{e^{(1-\theta)z}}{1-\theta z}$. Now, since $\frac{1}{1-\theta z} = \sum_{i \geq 0} (\theta z)^i$ and $e^{(1-\theta)z} = \sum_{i \geq 0} (1-\theta)^i z^i / i!$, we have that

$$\sum_{\sigma \in \mathbb{S}_M} \theta^{M-f(\sigma)} = M! \sum_{i=0}^M \frac{\theta^{M-i} (1-\theta)^i}{i!},$$

as required. \square

B. Computing V

Recall that $V \triangleq \sum_{\sigma, \tau \in \mathbb{S}_M} \text{Prob}(I(\sigma) \wedge I(\tau))$. As pointed out earlier, the quantity V comprises $(M!)^2 = \Theta(M(M/e)^{2M})$ summands and in this subsection, we borrow ideas from trellis theory [30] to reduce the running time to $O(M^{1.5}4^M)$.

Before we discuss about trellis theory, we recall the definition of the matrices \mathbf{T}_{BEC} , $\mathbf{T}_{\text{BSC}}^{(u)}$, and $\mathbf{T}_{\text{BSC}}^{(\ell)}$ and establish a relationship of the quantity V with the following permanent-like matrix function.

Definition 5. Let \mathbf{T} be an $M \times M$ -matrix whose (i, j) -th entry is T_{ij} . For $i, k \in [M]$, we further define $\phi(T_{ij}, T_{kj}) = T_{ij}$ if $i = k$ and $\phi(T_{ij}, T_{kj}) = T_{ij}T_{kj}$ if $i \neq k$. Then the *second-order permanent* of \mathbf{T} is given by the value

$$\text{per}^{(2)}(\mathbf{T}) \triangleq \sum_{\sigma, \tau \in \mathbb{S}_M} \prod_{j \in [M]} \phi(T_{\sigma(j)j}, T_{\tau(j)j}). \quad (9)$$

We have the following analogue of Proposition 10.

Proposition 12. Let \mathcal{C} be a code and let $0 < p < 1/2$. Recall that $V \triangleq \sum_{\sigma, \tau \in \mathbb{S}_M} \text{Prob}(I(\sigma) \wedge I(\tau))$.

(i) If we transmit the codebook \mathcal{C} through a BEC(p), then $V = \text{per}^{(2)}(\mathbf{T}_{\text{BEC}})$.

(ii) If we transmit the codebook \mathcal{C} through a BSC(p), then $\text{per}^{(2)}(\mathbf{T}_{\text{BSC}}^{(\ell)}) \leq V \leq \text{per}^{(2)}(\mathbf{T}_{\text{BSC}}^{(u)})$.

Proof. The proof is similar to the proof of Proposition 10. Hence, we only sketch the proof for the binary erasure channel. Fix a pair of permutations σ and τ . Then $I(\sigma) \wedge I(\tau)$ is the event that input-output graph \mathcal{G} contains both perfect matchings σ and τ . Hence, for all $j \in [M]$, if $\sigma(j) \neq \tau(j)$, we have both edges: one edge between the input x_j and the output of $x_{\sigma(j)}$, and another edge between the input x_j and the output of $x_{\tau(j)}$. If $\sigma(j) = \tau(j)$, we have the edge between the input x_j and the output of $x_{\sigma(j)}$. To conclude the proof, we then proceed as in the proof of Proposition 10(i). \square

Therefore, to compute or estimate V , we determine the second-order permanents of \mathbf{T}_{BEC} , $\mathbf{T}_{\text{BSC}}^{(u)}$, and $\mathbf{T}_{\text{BSC}}^{(\ell)}$. To do so, we adapt techniques from our recent work [29], where we proposed a very different approach to exact permanent computation. In this subsection, we replicate and modify certain parts from [29] for our computation task. Specifically, to compute the second-order permanents, we use a

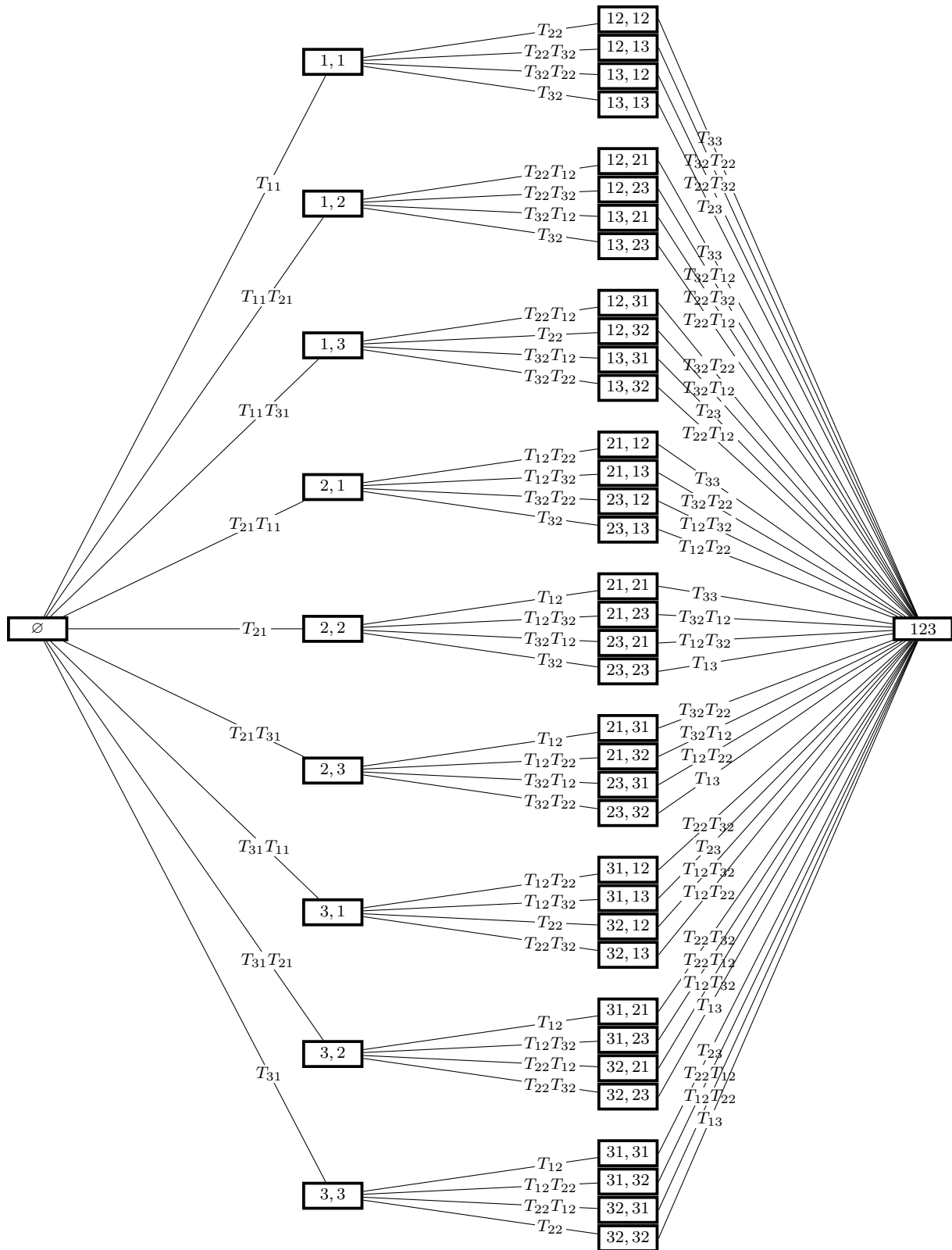


Fig. 1. Non-minimal trellis that computes $\text{per}^{(2)}(T)$ where T is a (3×3) -matrix. Here, the trellis has 47 vertices and 81 edges.

graph structure called *trellis*. The trellis was invented by Forney [33] over fifty years ago to illustrate the Viterbi decoding algorithm [34] for convolutional codes. It has since been studied extensively by coding theorists; see [30] for an excellent survey.

A *trellis* $\mathbb{T} = (\mathbb{V}, \mathbb{E}, \mathbb{L})$ is an edge-labelled directed graph, where \mathbb{V} is the set of *vertices*, \mathbb{E} is the set of ordered pairs $(v, v') \in \mathbb{V} \times \mathbb{V}$, called *edges*, and \mathbb{L} is the *edge-labelling function*. Specifically, \mathbb{L} is a real-valued function that maps an edge $e \in \mathbb{E}$ to a real number. The defining property of a trellis is that the set \mathbb{V} of vertices can be partitioned into $\mathbb{V}_0, \mathbb{V}_1, \dots, \mathbb{V}_M$ such that every edge (v, v') begins at $v \in \mathbb{V}_{j-1}$ and terminates at $v' \in \mathbb{V}_j$ for some $j \in [M]$. In addition, the subsets \mathbb{V}_0 and \mathbb{V}_M are singletons, containing two distinguished vertices, called the *root* and the *toor*, respectively.

For each path \mathbf{p} defined by its edge sequence $e_1 e_2 \dots e_t$, we associate the path \mathbf{p} with its *value* $\text{val}(\mathbf{p}) \triangleq \prod_{j=1}^t \mathbb{L}(e_j)$. Furthermore, we use $\mathbb{P}(\mathbb{T})$ to denote the *multiset* of all paths from the root to toor and we are interested in the *value* of the trellis, defined by $\text{val}(\mathbb{T}) \triangleq \sum_{\mathbf{p} \in \mathbb{P}(\mathbb{T})} \text{val}(\mathbf{p})$. Then the celebrated Viterbi algorithm¹ is an application of the dynamic programming method that computes $\text{val}(\mathbf{p})$ efficiently. Specifically, the Viterbi algorithm computes $\text{val}(\mathbb{T})$ using exactly $|\mathbb{E}| - \deg(\text{root})$ multiplications and exactly $(|\mathbb{E}| - |\mathbb{V}| + 1)$ additions.

Example 6. Set $M = 3$ and we consider a 3×3 -matrix \mathbf{T} . Then we can use the trellis \mathbb{T} in Figure 1 to compute $\text{per}^{(2)}(\mathbf{T})$. In particular, $\text{val}(\mathbb{T}) = \text{per}^{(2)}(\mathbf{T})$. We can readily check that $|\mathbb{V}_0| = |\mathbb{V}_3| = 1$, $|\mathbb{V}_1| = 9$ and $|\mathbb{V}_2| = 36$ and so, $|\mathbb{V}| = 47$ and $|\mathbb{E}| = 81$. Therefore, the Viterbi algorithm computes $\text{val}(\mathbb{T}) = \text{per}^{(2)}(\mathbf{T})$ using 72 multiplications and 35 additions.

Consider some trellis \mathbb{T} . Since the complexity of evaluating the value of \mathbb{T} depends on its number of vertices and edges, one key objective in the study of trellises in coding theory is to find a “smaller” trellis \mathbb{T}' so that $\mathbb{P}(\mathbb{T}') = \mathbb{P}(\mathbb{T})$. Formally, we say that $\mathbb{T}^* = (\mathbb{V}^*, \mathbb{E}^*, \mathbb{L}^*)$ is a *minimal trellis* for \mathbb{T} if $\mathbb{P}(\mathbb{T}^*) = \mathbb{P}(\mathbb{T})$ and the following holds:

for all other trellises $\mathbb{T}' = (\mathbb{V}, \mathbb{E}, \mathbb{L})$ such that $\mathbb{P}(\mathbb{T}') = \mathbb{P}(\mathbb{T})$, we have that $|\mathbb{V}_j^*| \leq |\mathbb{V}_j|$ for all $j = 0, 1, \dots, M$.

There are examples of trellises that do not admit a minimal trellis representation. Nevertheless, if the \mathbb{T} obeys certain properties, we have that \mathbb{T} admits a unique minimal trellis. Moreover, there is a simple *merging* procedure that finds this trellis [35], [36]. Here, we omit the details of the merging process. Instead, we simply describe the minimal trellis \mathbb{T}_M^* that computes $\text{per}^{(2)}(\mathbf{T})$ for some given matrix \mathbf{T} .

Definition 7 (Trellis for Second-Order Permanents). Fix M and let \mathbf{T} be an $M \times M$ -matrix. Then the *second-order-permanent trellis* \mathbb{T}_M^* is defined as follows.

- (Vertices) For $0 \leq j \leq M$, we set

$$\begin{aligned} \mathbb{V}_j^{(1)} &\triangleq \{X \subset [M] : |X| = j\}, \\ \mathbb{V}_j^{(2)} &\triangleq \{\{X_1, X_2\} : X_1, X_2 \subseteq [M], |X_1| = |X_2| = j\}. \end{aligned}$$

and let $\mathbb{V}_j^{(1)} = \mathbb{V}_j \cup \mathbb{V}_j^{(2)}$. In other words, we have two types of vertices: the first type comprises all $\binom{M}{j}$ j -subsets of $[M]$; while the second type comprises all $\binom{M}{2}$ unordered pairs of j -subsets.

¹We omit a detailed description of the Viterbi algorithm here and instead, refer the interested reader to the meticulous exposition in [30]. The original algorithm was introduced by Viterbi [34] in 1967 to perform maximum-likelihood decoding of convolutional codes. In [29], we described the Viterbi algorithm in the context of permanent computations.

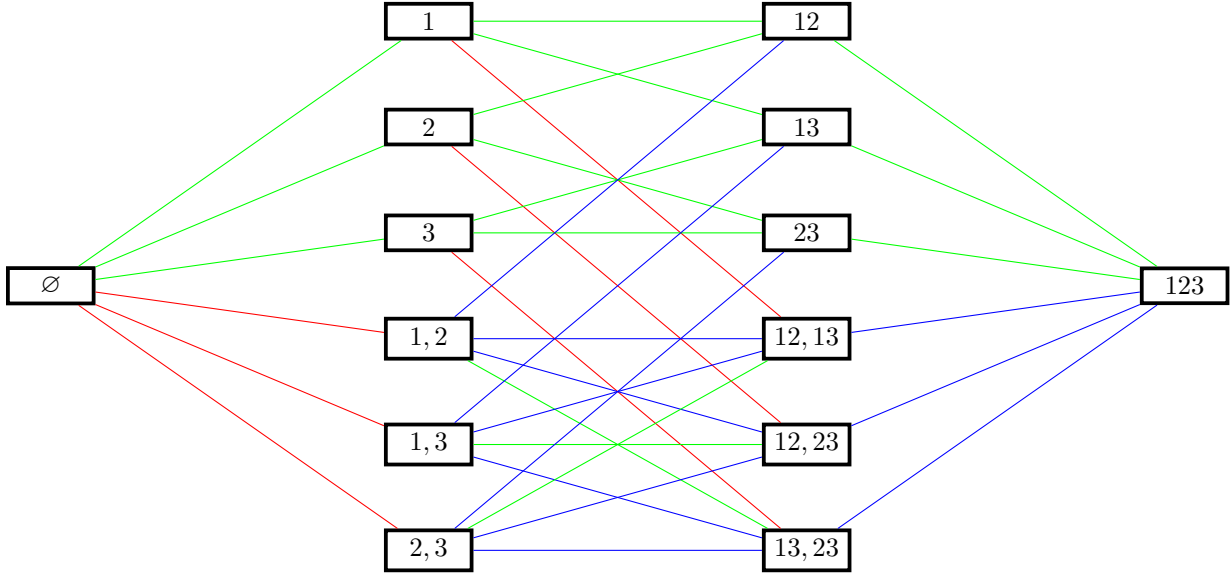


Fig. 2. Minimal trellis \mathbb{T}_3^* that computes $\text{per}^{(2)}(T)$ where T is a (3×3) -matrix. To reduce clutter, edge labels have been removed. Instead, green, blue, and red edges denote edges with labels of the form T_{ij} , $T_{ij}T_{kj}$, and $2T_{ij}T_{kj}$, respectively. We refer the reader to Definition 7 for details of the labelling. Here, \mathbb{T}_3^* has 14 vertices and 33 edges.

- (Edges and edge labels) First, we consider $X \in \mathbb{V}_{j-1}^{(1)}$ and so, $|X| = j - 1$.
 - For $i \notin X$, we draw the edge from X to $X \cup \{i\}$ and we label this edge with T_{ij} .
 - For $i, k \notin X$ and $i < k$, we draw an edges $X \rightarrow \{X \cup \{i\}, X \cup \{k\}\}$, and we label this edge with $2T_{ij}T_{kj}$.

Next, we consider $\{X_1, X_2\} \in \mathbb{V}_{j-1}^{(2)}$ and so, $|X_1| = |X_2| = j - 1$.

- Suppose that there exists $|X_3| = j$ such that $X_3 = X_1 \cup X_2$. Then we draw an edge from $\{X_1, X_2\}$ to X_3 and label this edge with $T_{ij}T_{kj}$, where $\{i\} = X_3 \setminus X_1$ and $\{k\} = X_3 \setminus X_2$.
- If $i \neq k$, $i \notin X_1$ and $k \notin X_2$, we draw an edge from $\{X_1, X_2\}$ to $\{X_1 \cup \{i\}, X_2 \cup \{k\}\}$ and we label this edge with $T_{ij}T_{kj}$.
- If $i \notin X_1 \cup X_2$, we draw an edge from $\{X_1, X_2\}$ to $\{X_1 \cup \{i\}, X_2 \cup \{i\}\}$ and we label this edge with T_{ij} .

Example 8 (Example 6 continued). As before, set $M = 3$ and we consider a 3×3 -matrix T . Then second-order-permanent trellis \mathbb{T}_3^* in Figure 2. We check that \mathbb{T}_3^* has 14 vertices and 33 vertices. Hence, applying to the Viterbi algorithm on \mathbb{T}_3^* , we compute $\text{per}^{(2)}(T)$ using only 27 multiplications and 20 additions. The number of arithmetic operations is significantly lesser than the number in Example 6.

Proposition 13. Let $\mathbb{T}_M^* = (\mathbb{V}^*, \mathbb{E}^*, \mathbb{L}^*)$ be as defined by Definition 7. Then we have that

$$|\mathbb{V}^*| = \frac{1}{2} \binom{2M}{M} + 2^{M-1} \sim \frac{4^M}{2\sqrt{\pi M}} \quad (10)$$

$$|\mathbb{E}^*| \leq \frac{M^2}{2} \binom{2M-2}{M-1} + M(M+1)2^{M-3} \sim \frac{M^{1.5}4^{M-1}}{2\sqrt{\pi}}. \quad (11)$$

Proof. First, we determine the total number of vertices. Now, for all j , we have that $\mathbb{V}_j^{(1)} = \binom{M}{j}$ and therefore, $\left| \bigcup_{j=0}^M \mathbb{V}_j^{(1)} \right| = \sum_{j=0}^M \binom{M}{j} = 2^M$.

On the other hand, for all j , we have that $\mathbb{V}_j^{(2)} = \binom{(M)}{2}$. So,

$$\begin{aligned} \left| \bigcup_{j=0}^M \mathbb{V}_j^{(2)} \right| &= \sum_{j=0}^M \frac{\binom{(M)}{j} (\binom{(M)}{j} - 1)}{2} \\ &= \frac{1}{2} \sum_{j=0}^M \binom{(M)}{j}^2 - \binom{(M)}{j} \\ &= \frac{1}{2} \sum_{j=0}^M \binom{(M)}{j}^2 - \frac{1}{2} \sum_{j=0}^M \binom{(M)}{j} \\ &= \frac{1}{2} \binom{(2M)}{M} - 2^{M-1} \end{aligned}$$

Therefore, the total number of vertices is $\frac{1}{2} \binom{(2M)}{M} + 2^{M-1}$, as required.

Next, we determine the total number of edges. Now, for all j , a vertex in either $\mathbb{V}_j^{(1)}$ or $\mathbb{V}_j^{(2)}$ has out-degree at most $(M-j)^2$. Therefore, the total number of edges exiting vertices in $\bigcup_{j=0}^M \mathbb{V}_j^{(1)}$ is at most

$$\sum_{j=0}^M (M-j)^2 \binom{(M)}{j} = M(M+1)2^{M-2}.$$

On the other hand, the total number of edges exiting vertices in $\bigcup_{j=0}^M \mathbb{V}_j^{(2)}$ is at most

$$\begin{aligned} \sum_{j=0}^M (M-j)^2 \binom{\binom{(M)}{2}}{j} &= \frac{1}{2} \sum_{j=0}^M (M-j)^2 \left(\binom{(M)}{j}^2 - \binom{(M)}{j} \right) \\ &= \frac{M^2}{2} \binom{(2M-2)}{M-1} - M(M+1)2^{M-3}. \end{aligned}$$

Therefore, we have at most $\frac{M^2}{2} \binom{(2M-2)}{M-1} + M(M+1)2^{M-3}$ edges, as required. Throughout this proof, we have used a number of combinatorial identities obtained from [37]–[39] (detailed proofs can be found in the references therein). \square

Therefore, we have the following method that computes $\text{per}^{(2)}(\mathbf{T})$ in $O(M^{1.5}4^{M-1}) = o((M!)^2)$ time.

Corollary 14. *Let \mathbf{T} be an $M \times M$ -matrix. We can compute $\text{per}^{(2)}(\mathbf{T})$ in $O(M^{1.5}4^{M-1})$ time using the trellis \mathbb{T}_M^* .*

C. Simulation Results

To end this section, we corroborate our estimates given by (3), (4) and Theorem 11 with numerical experiments. Specifically, we consider the binary erasure channel BEC(p) and estimate $P_{\text{error}}(\mathcal{C})$, where \mathcal{C} are the codes defined in Examples 1 and 3. Then using the methods described in Section V-B, we determine the values of U and V , and hence, obtain upper and lower bounds for $P_{\text{error}}(\mathcal{C})$ using (3) and (4), respectively. We also compute the upper bound provided by the closed formula in Theorem 11. We then simulated 500,000 trials for various erasure probabilities p and determined numerically the average failure rate. The results are plotted in Figure 3 and we observe that the estimates provided by (3) and (4) are very sharp.

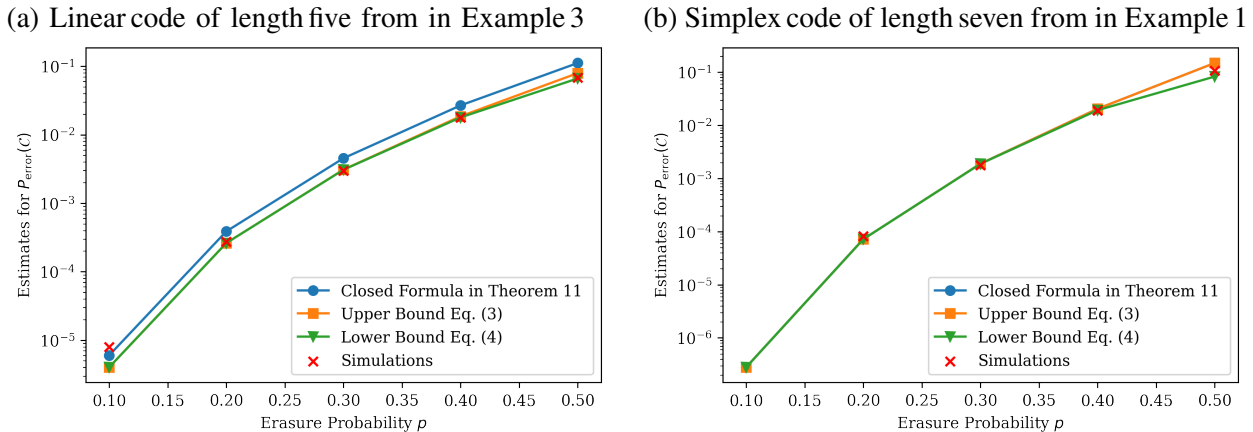


Fig. 3. Estimates for the probability of erroneous identification $P_{\text{error}}(\mathcal{C})$ for various codes \mathcal{C}

VI. DISCUSSION AND FUTURE WORK

We discuss certain extensions and possible future work.

- *General channels.* As mentioned earlier, both JEDI and JMDI can be extended to obtain a joint maximum likelihood decoder for other channels. Specifically, suppose that a channel is described by a probability distribution P where each output \mathbf{y} given an input \mathbf{x} is received with probability $P(\mathbf{y}|\mathbf{x})$. In Step 1 of JEDI / JMDI, we can create the bipartite input-output graph \mathcal{G} by drawing an edge (\mathbf{x}, \mathbf{y}) whenever $P(\mathbf{y}|\mathbf{x}) > 0$, and then assigning the edge (\mathbf{x}, \mathbf{y}) the cost $-\log P(\mathbf{y}|\mathbf{x})$. Then finding a minimum-cost perfect matching in the \mathcal{G} yields a permutation that maximizes the likelihood of correct identification. As with the analysis of JEDI, the time to find a minimum-cost perfect matching depends on the size of \mathcal{G} , or the number of edges in \mathcal{G} . In [12], Chrisnata *et al.* determined the expected number of edges in \mathcal{G} for both the insertion and deletion channels. It will be interesting to study this quantity for other channels.
- *Handling absentee bees.* In [11], the authors studied the bee-identification problem for the scenario where bees were absent with certain probability. In other words, instead of M channel outputs, we have $M - a$ outputs where $a > 0$. Both JEDI and JMDI can be modified to handle these scenarios. In both cases, we proceed as before and simply add a absentee right nodes to the bipartite graph \mathcal{G} . For the BEC case, we connect each absentee right node to all left nodes, while for the BSC case, we connect each absentee right node to all left nodes and assign the cost to be zero. Then we find a perfect matching or a minimum-cost perfect matching as before.
- *Code Design.* In this paper, given a code \mathcal{C} , we can construct the matrices T_{BEC} and $T_{\text{BSC}}^{(u)}$ as in Section V, and then estimate certain performance metrics of JEDI and JMDI. Specifically, the sum of entries of T_{BEC} determines the running time of JEDI, while the permanents of T_{BEC} and $T_{\text{BSC}}^{(u)}$ provide upper bounds on the probability of erroneous identification. Alternatively, we can fix the probability of erroneous identification ϵ , and design a code \mathcal{C} such that $P_{\text{error}}(\mathcal{C}) \leq \epsilon$. This was partially studied in [2]. Specifically, for fixed ϵ , Tandon *et al.* showed that there exists a random code \mathcal{C} with rate close to $R(\epsilon)$ and $P_{\text{error}}(\mathcal{C}) \leq \epsilon$. A natural question is to find an *explicit* family of codes that achieve the same property.

REFERENCES

- [1] Kiah, H. M., Vardy, A., and Yao, H. (2021, July). Efficient Bee Identification. In 2021 IEEE International Symposium on Information Theory (ISIT) (pp. 1943-1948). IEEE.
- [2] A. Tandon, V. Y. F. Tan, and L. R. Varshney, "The Bee-Identification Problem: Bounds on the Error Exponent," *IEEE Trans. Commun.*, vol. 67, no. 11, pp. 7405–7416, 2019.
- [3] A. B. Poore and S. Gadaleta, "Some assignment problems arising from multiple target tracking," *Math. Comput. Modeling*, vol. 43, nos. 9–10, pp. 1074–1091, 2006.
- [4] T. Gernat, V. D. Rao, M. Middendorf, H. Dankowicz, N. Goldenfeld, and G. E. Robinson, "Automated monitoring of behavior reveals bursty interaction patterns and rapid spreading dynamics in honeybee social networks," *Proc. Nat. Acad. Sci. USA*, vol. 115, no. 7, pp. 1433–1438, Feb. 2018.
- [5] A. Pananjady, M. J. Wainwright, and T. A. Courtade, "Linear regression with shuffled data: Statistical and computational limits of permutation recovery," *IEEE Trans. Inf. Theory*, vol. 64, no. 5, pp. 3286–3300, May 2018.
- [6] J. L. Schmid-Burgk, R. M. Schmithausen, D. Li, R. Hollstein, A. Ben-Shmuel, O. Israeli, S. Weiss, N. Paran, G. Wilbring, J. Liebing, D. Feldman, M. Ślabicki, B. Lippke, E. Sib, J. Borrajo, J. Strecker, J. Reinhardt, P. Hoffmann, B. Cleary, M. Hölzel, M. M. Nöthen, M. Exner, K. U. Ludwig, A. Regev, F. Zhang, "LAMP-Seq: Population-Scale COVID-19 Diagnostics Using Combinatorial Barcoding", *bioRxiv preprint 2020.04.06.025635*, 2020.
- [7] J. Li, W. Quan, S. Yan, S. Wu, J. Qin, T. Yang, F. Liang, D. Wang, Y. Liang, "Rapid detection of SARS-CoV-2 and other respiratory viruses by using LAMP method with Nanopore Flongle workflow", *bioRxiv preprint 2020.06.03.131474*, 2020.
- [8] P. James, D. Stoddart, E. D Harrington, J. Beaulaurier, L. Ly, S. W. Reid, D. J Turner and S. Juul, "LamPORE: rapid, accurate and highly scalable molecular screening for SARS-CoV-2 infection, based on nanopore sequencing", *medRxiv preprint 2020.08.07.20161737*, 2020.
- [9] L. Peto, G. Rodger, D. P. Carter, K. L. Osman, M. Yavuz, K. Johnson, M. Raza, M. D. Parker, M. D Wyles, M. Andersson, A. Justice, A. Vaughan, S. Hoosdally, N. Stoesser, P. C Matthews, D. W Eyre, T. EA Peto, M. W Carroll, T. I de Silva, D. W Crook, C. M Evans, S. T Pullan, "Diagnosis of SARS-CoV-2 infection with LamPORE, a high-throughput platform combining loop-mediated isothermal amplification and nanopore sequencing", *medRxiv preprint 2020.09.18.20195370*, 2020.
- [10] A. S. Boeshaghi, N. B. Lubock, A. R. Cooper, S. W. Simpkins, J. S. Bloom, J. Gehring, L. Luebbert, S. Kosuri, L. Pachter, "Reliable and accurate diagnostics from highly multiplexed sequencing assays." *Sci Rep* 10, 21759, 2020.
- [11] A. Tandon, V. Y. F. Tan, and L. R. Varshney, "The Bee-Identification Error Exponent With Absentee Bees," *IEEE Trans. Inform. Theory*, vol. 66, no. 12, pp. 7602–7614, 2020.
- [12] Chrisnata, J., Kiah, H. M., Vardy, A., and Yaakobi, E. (2022, June). "Bee Identification Problem for DNA Strands". In 2022 IEEE International Symposium on Information Theory (ISIT) (pp. 969-974). IEEE.
- [13] J. E. Hopcroft, R. M. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs", *SIAM Journal on Computing*, 2 (4), pp. 225–231, 1973.
- [14] H. W. Kuhn, "The Hungarian Method for the assignment problem", *Naval Research Logistics Quarterly*, 2, pp. 83–97, 1955.
- [15] K. Fukuda, and T. Matsui, "Finding all the perfect matchings in bipartite graphs," *Applied Mathematics Letters*, 7(1), pp. 15–18, 1954.
- [16] T. M. Hoang, T. Thierauf, M. Mahajan, "On the bipartite unique perfect matching problem," In *ICALP 2006. LNCS*, vol. 4051, pp. 453–464. Springer, Heidelberg.
- [17] V. V. Zyablov and M. S. Pinsker, "Estimation of the error-correction complexity of Gallager low-density codes," *Problems of Information Transmission*, 11(1), pp. 18–28, 1976
- [18] F. J. MacWilliams, and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.
- [19] I. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *Trans. IRE Professional Group Inform. Theory*, vol. 4, no. 4, pp. 38–49, 1954.
- [20] R. R. Green, "A serial orthogonal decoder," *JPL Space Programs Summary*, vol. 37, pp. 247–253, 1966.
- [21] Y. Be'ery and J. Snyders, "Optimal soft decision block decoders based on fast Hadamard transform," *IEEE Trans. Inform. Theory*, vol. 32, no. 3, pp. 355–364, 1986.
- [22] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *Journal. ACM*, 19, 2, pp. 248–264, 1972
- [23] N. Tomizawa, "On some techniques useful for solution of transportation network problems," *Networks*, 1, 2, pp. 173–194, 1971.

- [24] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005
- [25] A. Bhowmick and S. Lovett, "The List Decoding Radius for Reed–Muller Codes Over Small Fields," *IEEE Trans. Inform. Theory*, vol. 64, no. 6, pp. 4382–4391, 2020.
- [26] Valiant, L. G. (1979). The Complexity of Computing the Permanent. *Theoretical Computer Science*, 8 (2): 189–201.
- [27] Nijenhuis, A. and Wilf, H. S. (1978). *Combinatorial algorithms: for computers and calculators*. Academic press.
- [28] Glynn, D. G. (2010). The permanent of a square matrix. *European Journal of Combinatorics*, 31(7):1887–1891.
- [29] Kiah, H. M., Vardy, A., and Yao, H. (2021). Computing Permanents on a Trellis. arXiv preprint arXiv:2107.07377.
- [30] Vardy, A. (1998) Trellis structure of codes, Ch. 24, pp. 1989–2118, in the *Handbook of Coding Theory*, edited by V. S. Pless and W. C. Huffman, Amsterdam: North-Holland/Elsevier.
- [31] Barg, A., and Forney, G. D. (2002). Random codes: Minimum distances and error exponents. *IEEE Transactions on Information Theory*, 48(9), 2568-2573.
- [32] Sedgewick, R., and Flajolet, P. (2013). *An introduction to the analysis of algorithms*. Pearson Education India.
- [33] Forney, G. D. Jr. (1967). Final report on a coding system design for advanced solar missions. Contract NAS2-3637, NASA Ames Research Center, CA, December.
- [34] Viterbi, A. J. (1967) Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions Information Theory*, vol. 13, pp. 260–269.
- [35] Kschischang, F. R. (1996). The trellis structure of maximal fixed cost codes. *IEEE Transactions Information Theory*, vol. 42, pp. 1828–1838, 1996.
- [36] Vardy, A. and Kschischang, F. R. (1996). Proof of a conjecture of McEliece regarding the expansion index of the minimal trellis. *IEEE Transactions Information Theory*, pp. 2027–2033, 1996.
- [37] OEIS Foundation Inc. (2022), Entry A037966 in The On-Line Encyclopedia of Integer Sequences, <http://oeis.org/A037966>
- [38] OEIS Foundation Inc. (2022), Entry A001788 in The On-Line Encyclopedia of Integer Sequences, <http://oeis.org/A001788>
- [39] OEIS Foundation Inc. (2022), Entry A000984 in The On-Line Encyclopedia of Integer Sequences, <http://oeis.org/A000984>