Efficient Bee Identification

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: hmkiah@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.**

Here, we study efficient ways of performing joint decoding. First, 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.

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 [1]. Besides problems involving multiple target tracking [2], [3], the bee-identification problem is also relevant to other applications (see [1], [4] for other examples). One recent possible application is that of pooled testing for viral RNA like COVID-19. In a recent experiment [5], 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 [6]–[9].

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 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 [1], 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 [10].

In [1], 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 [11] and Hungarian method [12], 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.

We formally state the problem and our contributions in the next section. For the ease of exposition, we study the bee-identification problem for the binary erasure channel (BEC) and binary symmetric channel (BSC). The algorithms described in the paper can be extended for larger alphabets and to perform joint maximum-likelihood decoding for other channels like the discrete memoryless and deletion channels.

II. Problem Definition

Consider a binary code $\mathcal{C}$ of length n with M *codewords* $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$. Consider, in addition, a binary channel where each output $\boldsymbol{y}$ given an input $\boldsymbol{x}$ is received with probability $P(\boldsymbol{y}|\boldsymbol{x})$. We send *all* M codewords over the channel and obtain an unordered set of M outputs $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_M\}$. Note that $\boldsymbol{y}_i$ is not necessarily the channel output of $\boldsymbol{x}_i$ and in fact, the task of the *bee-identification problem* is to find a length-M permutation π so that $\boldsymbol{y}_{\pi(i)}$ is indeed the channel output of the input $\boldsymbol{x}_i$ for all $i \in [M] \triangleq \{1, 2, \ldots, M\}$. Assuming the channels are independent, the *joint decoder* finds a length-M permutation π that maximizes the probability $\prod_{i\in[M]} P(\boldsymbol{y}_{\pi(i)}|\boldsymbol{x}_i)$.

In this paper, we study efficient ways of determining the permutation π. Since the input to our problem are 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, and hence, in most parts of the paper, we suppress factors involving n in the big-O notation. We note that this is different from the setting in [1] where $M = 2^{Rn}$ for some rate R. Nevertheless, many error estimates derived in the latter remain applicable. The focus of this work is to find efficient identifier algorithms that achieve these error probabilities.

Our Contributions.

- For the BEC with erasure probability p, 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 with crossover probability p, we provide a joint decoder, Joint Minimum-Distance Decoding Identifier (JMDI), 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 p, JLDI terminates in $O(M^{2+\epsilon})$ time for any small ϵ and is almost as good as JMDI (see Theorem 10 for the formal statement).

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 [1], 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 $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$ and $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_M$, respectively. Suppose that we have a perfect matching $\mathcal{M}$ of $\mathcal{G}$. Then we can write the edges of $\mathcal{M}$ as $(x_1, y_{\pi(1)}), (x_2, y_{\pi(2)}), \ldots, (x_M, y_{\pi(M)})$ and it follows from the definition of a matching that π 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 π, we obtain a perfect matching of $\mathcal{G}$ if $(\boldsymbol{x}_i, \boldsymbol{y}_{\pi(i)})$ 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 $\mathcal{C} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\} \subseteq \{0,1\}^n$ of size M and a set of M channel outputs $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_M\} \subseteq \{0,1,?\}^n$.

Output: A permutation π such that $\boldsymbol{y}_i$ matches $\boldsymbol{x}_{\pi(i)}$ for all $i \in [M]$ if there is a unique π. 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 $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ if and only if $\boldsymbol{y}_j$ matches $\boldsymbol{x}_i$. Here, $\boldsymbol{y}$ *matches* $\boldsymbol{x}$ if both $\boldsymbol{y}$ coincides with $\boldsymbol{x}$ on positions that are not erased.
(2) Determine if there is a unique perfect matching in $\mathcal{G}$. If the matching π is unique, return π. If the matching is not unique, return Failure.

We discuss the running time of JEDI. For general codebooks, Step 1 can be implemented in $O(M^2)$ time.

Next, let the bipartite graph constructed in Step 1 be $\mathcal{G}$. For each codeword $\boldsymbol{x} \in \mathcal{C}$, let $Y(\boldsymbol{x})$ be its corresponding channel output and we have that $Y(\boldsymbol{x})$ matches $\boldsymbol{x}$. Therefore, the set of edges $\{(\boldsymbol{x}, Y(\boldsymbol{x})) : \boldsymbol{x} \in \mathcal{C}\}$ is a perfect matching of $\mathcal{G}$ and hence, the number of edges is at least M. For Step 2, we can use the Hopcraft-Karp algorithm [11] to find a perfect matching in $O(E\sqrt{M})$ time. Following the methods described in Fukada [13] and Hoang *et al.* [14], we then determine if another perfect matching exists in $O(M+E) = O(E)$ time. Hence, Step 2 can be implemented in $O(E\sqrt{M}) = O(M^{2.5})$ time. Therefore, a simple analysis shows that JEDI runs in $O(M^{2.5})$ time.

First, we *improve 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 [15]. 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 the 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) and thus, we can terminate our search. 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.

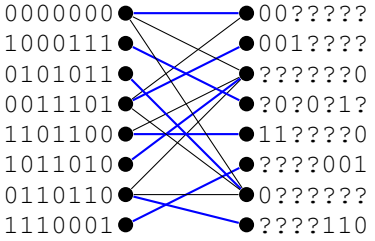
(1) Initialize $\mathcal{M}$ to the empty set.
(2) Find a node u in $\mathcal{G}$ with degree one. Here, u may be a left *or* right node. If there is no such node, go to Step 6.
(3) Let uv be the unique edge incident to u and add uv to $\mathcal{M}$.
(4) Remove nodes u and v and all edges incident to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

00?????, 001????, ??????0, ?0?0?1?,
11????0, ????001, 0??????, ????110.

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 $\mathcal{G}$.



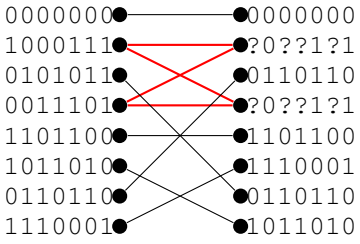
Here, we list the edges *in the order* they are added to $\mathcal{M}$ according to PMA.

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

(b) Suppose the channel outputs are:

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

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 [15]. For completeness, we provide a detailed proof in Section III-B.

Lemma 2. *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 [16]). 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 d. 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 3. *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 $\boldsymbol{x}$ and $\boldsymbol{z}$ and let $\tilde{\boldsymbol{x}}$ be the channel output of $\boldsymbol{x}$. We first compute the probability that there is an edge between the nodes $\boldsymbol{z}$ and $\tilde{\boldsymbol{x}}$ in JEDI. Suppose the distance between $\boldsymbol{x}$ and $\boldsymbol{z}$ is d. Then there is an edge between $\boldsymbol{z}$ and $\tilde{\boldsymbol{x}}$ if the d bits where $\boldsymbol{x}$ and $\boldsymbol{z}$ differ are erased. In other words, there is an edge between $\boldsymbol{z}$ and $\tilde{\boldsymbol{x}}$ with probability p^d. Since expectation is linear, the expected number of edges is $\sum_{\boldsymbol{x},\boldsymbol{z}\in\mathcal{C}} z^{d_H(\boldsymbol{x},\boldsymbol{z})} = B(z)$. Here, $d_H(\boldsymbol{x},\boldsymbol{z})$ denotes the Hamming distance of $\boldsymbol{x}$ and $\boldsymbol{z}$. □

Consider a code $\mathcal{C}$ with minimum distance d. Then we have that $B_1 = B_2 = \cdots = B_{d-1} = 0$. 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 of $\mathcal{G}$ is at most $M + \binom{M}{2} p^d$.

So, it remains to improve the running time of Step 1, i.e. the time to set up the graph $\mathcal{G}$. To do so, we set $\mathcal{C}$ to be a linear code of length n. Then for any channel output $\boldsymbol{y}$, we can find all $\boldsymbol{x}$ such that $\boldsymbol{y}$ matches $\boldsymbol{x}$ in $O(n^3)$ time by matrix inversion. Therefore, we can construct $\mathcal{G}$ in $O(Mn^3)$ time. Hence, we summarize our discussion with the following theorem.

Theorem 4. *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 $O\left(M + \binom{M}{2} p^d + Mn^3\right)$.*

Suppose that $\mathcal{C}_n$ 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 $\binom{M_n}{2} p^d = o(1)$, then the expected running time of JEDI tends to $O(Mn^3)$.

A. Reed-Muller Codes

In this subsection, we apply Theorem 4 to the ubiquitous class of Reed-Muller codes [17] and derive the expected running time of JEDI on this class of linear codes.

Theorem 5. *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 4, it remains to show that $\binom{M}{2} p^d$ tends to zero as $m \to \infty$, or equivalently,

$$\log \binom{M}{2} + d \log p \to -\infty. \tag{1}$$

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.

Next, we consider a small constant positive ϵ.

- When $r \geq 2$, 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 $2^{3m} = O\left(2^{\gamma m^r}\right) = O(M^\epsilon)$.
- 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) [18], [19] to compute the edges. Specifically, for each channel output $\boldsymbol{y}$, we can use FHT to find all $\boldsymbol{x}$ that matches $\boldsymbol{y}$ 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. □

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 2.

Following Fukada [13] and Hoang *et al.* [14], 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 6. *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 6, 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 V).

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 [12].

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 $(\boldsymbol{x}, \boldsymbol{y})$, we connect them with an edge of cost $d_H(\boldsymbol{x}, \boldsymbol{y})$. Then the problem of finding a permutation π that maximizes the the probability $\prod_{i\in[M]} P(\boldsymbol{y}_{\pi(i)}|\boldsymbol{x}_i)$ 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 (JMDI).

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

Output: A permutation π such that the quantity $\sum_{i\in[M]} d_H(\boldsymbol{x}_i, \boldsymbol{y}_{\pi(i)})$ 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 $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ and set its cost to be $d_H(\boldsymbol{x}_i, \boldsymbol{y}_j)$.
(2) Find a minimum-cost matching in $\mathcal{G}$.

We discuss the running time of JMDI. In Step 1, we can compute the distance between all pair of words in $O(M^2n)$ time. In Step 2, we can apply the Hungarian method [12] and hence, we have the following theorem.

Theorem 7. *JMDI finds a permutation in $O(M^3 + M^2n)$ time.*

To improve the running time for the BSC channel, instead of finding the exact solution, we approximate it by finding 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 $\boldsymbol{y}$ is given by the number of codewords whose distance is at most R from $\boldsymbol{y}$. 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 $\boldsymbol{y} \in \{0,1\}^n$, we have that $|\{\boldsymbol{x} \in \mathcal{C} : d_H(\boldsymbol{x}, \boldsymbol{y}) \leq R\}|$ 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 [20], [21] and hence, JLDI terminates in $O(M^2(\log M + L + n))$ time.

Example 8. 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:

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

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 $(\boldsymbol{x}_i, \boldsymbol{y}_j)$, i.e. $d_H(\boldsymbol{x}_i, \boldsymbol{y}_j)$. We refer to this table as the cost matrix of $\mathcal{G}$.

Cost Matrix of $\mathcal{G}$:

	$\boldsymbol{y}_1$	$\boldsymbol{y}_2$	$\boldsymbol{y}_3$	$\boldsymbol{y}_4$
$\boldsymbol{x}_1 = 00000$	1	4	2	2
$\boldsymbol{x}_2 = 11100$	2	1	5	3
$\boldsymbol{x}_3 = 00111$	4	3	1	3
$\boldsymbol{x}_4 = 11011$	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 $\{(\boldsymbol{x}_i, \boldsymbol{y}_i) : i \in [4]\}$.

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.

Cost Matrix of $\mathcal{H}$:

	$\boldsymbol{y}_1$	$\boldsymbol{y}_2$	$\boldsymbol{y}_3$	$\boldsymbol{y}_4$
$\boldsymbol{x}_1 = 00000$	1	–	2	2
$\boldsymbol{x}_2 = 11100$	2	1	–	–
$\boldsymbol{x}_3 = 00111$	–	–	1	–
$\boldsymbol{x}_4 = 11011$	–	2	2	2

Indeed, we observe that the degree of $\boldsymbol{y}_i$, $i \in [4]$, 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 are rare and we estimate the probabilities.

Theorem 9. *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, define the event*

$$\textit{JMDI correctly finds } \pi \implies \textit{JLDI correctly finds } \pi\,. \quad (2)$$

Then we have that the probability event (2) occurs is at least $\left(1 - \exp(-\gamma^2 pn/3)\right)^M$.

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

To derive the probability estimates of event (2), we consider the random variable Z_i that denotes the number of errors for the codeword $\boldsymbol{x}_i$, $i \in [M]$. In other words, $Z_i = d_H(\boldsymbol{x}_i, \boldsymbol{y}_{\pi(i)})$. 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 $Z_i \le R$ 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 π. Also, for all i, since $Z_i \le R$, we have the edge $(\boldsymbol{x}_i, \boldsymbol{y}_{\pi(i)})$ 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 π. Hence, JLDI is correct.

Next, we lower bound the probability that all values of Z_i is at most R. Fix i. Using Chernoff's bound (see for example, [22]), we have that $P(Z_i \ge R) = P(Z_i \ge (1+\gamma)pn) \le \exp(-\gamma^2 pn/3)$ and so, $P(Z_i \le R) \ge 1 - \exp(-\gamma^2 pn/3)$. Since the values of Z_i are independent of each other, the lower bound follows. □

A. Reed-Muller Codes

Similar to before, we verify that the class of Reed-Muller codes satisfy the conditions of Theorem 9.

Theorem 10. *Fix $r \ge 1$ 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 11 (Bhomick and Lovett [23])**.** *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 10. Choose some small α so that $p < 1/2^r - \alpha$ and set $R = (1/2^r - \alpha)n$. Then Theorem 11 states that the list size is upper bounded by a constant independent of n and M. Hence, applying Theorem 9, we have the running time is $O(M^2(\log M + L + n)) = O(M^2(\log M + n))$.

When $r \ge 2$, we have that $n = O(M^\epsilon)$ as in the proof of Theorem 5 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 $\boldsymbol{y}$, we use FHT to compute $d_H(\boldsymbol{x}, \boldsymbol{y})$ for all codewords $\boldsymbol{x}$ 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 9 also states that event (2) occurs with probability at least $\theta_n \triangleq \left(1 - \exp(-\gamma^2 pn/3)\right)^M$ for some constant γ. Recall that M is the code size $2^{\sum_{i=0}^{r} \binom{m}{i}} \le 2^{(r+1)m^r} = O(2^{\lambda n})$ 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

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

□

V. Discussion and Future Work

We discuss certain extensions and possible future work.

- *General channels*. As mentioned earlier, JMDI can be modified and be used as joint (maximum likelihood) decoder for other channels. Specifically, suppose that a channel is described by a probability distribution P where each output $\boldsymbol{y}$ given an input $\boldsymbol{x}$ is received with probability $P(\boldsymbol{y}|\boldsymbol{x})$. We modify Step 1 in JMDI by assigning the edge $(\boldsymbol{x}, \boldsymbol{y})$ with the cost $-\log P(\boldsymbol{y}|\boldsymbol{x})$. Then finding a minimum-cost perfect matching in the resulting bipartite graph yields a permutation that maximizes the likelihood of correct identification.
- *Handling absentee bees*. In [10], 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.
- *Probability of Erroneous Identification*. In Theorems 5 and 10, we demonstrated that JMDI and JLDI for Reed-Muller codes terminates in almost linear and quadratic time, respectively. It remains to investigate the probability that the JMDI or JLDI recovers an incorrect permutation, or specifically, how fast the probability of erroneous identification decays to zero. Even though Tandon *et al.* [1] studied the joint-decoding error exponents for the case where codes have positive rates, their techniques appear relevant to general codes and it is interesting to how their results extend to zero-rate codes or other explicit families of codes.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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. Słabicki, 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.

[6] 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.

[7] 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.

[8] 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.

[9] A. S. Booeshaghi, 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.

[10] 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.

[11] 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.

[12] H. W. Kuhn, "The Hungarian Method for the assignment problem", *Naval Research Logistics Quarterly*, 2, pp. 83–97, 1955.

[13] K. Fukuda, and T. Matsui, "Finding all the perfect matchings in bipartite graphs," *Applied Mathematics Letters*, 7(1), pp. 15–18, 1954.

[14] 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.

[15] 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

[16] F. J. MacWilliams, and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[17] 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.

[18] R. R. Green, "A serial orthogonal decoder," *JPL Space Programs Summary*, vol. 37, pp. 247–253, 1966.

[19] 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.

[20] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *Journal. ACM*, 19, 2, pp. 248–264, 1972

[21] N. Tomizawa, "On some techniques useful for solution of transportation network problems," *Networks*, 1, 2, pp. 173–194, 1971.

[22] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005

[23] 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.