Bee Identification Problem for DNA Strands

Johan Chrisnata,[*‡] Han Mao Kiah,[*] Alexander Vardy,[†] and Eitan Yaakobi[‡]

[*] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
[†] Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093, USA
[‡] Department of Computer Science, Technion — Israel Institute of Technology, Haifa, 32000 Israel
{johanchr001,hmkiah}@ntu.edu.sg.edu, avardy@ucsd.edu, yaakobi@cs.technion.ac.il

***Abstract*—Motivated by DNA-based applications, we generalize the *bee identification problem* proposed by Tandon *et al.* (2019). In this setup, we transmit all M codewords from a codebook over some channel and each codeword results in N noisy outputs. Then our task is to identify each codeword from the MN noisy outputs.**

First, via a reduction to a minimum-cost flow problem on a related flow network $\mathcal{G}_N$, we show that the problem can be solved in $O(M^3)$ time in the worst case. Next, we consider the deletion channel and study the expected number of edges in the network $\mathcal{G}_N$. Specifically, we obtain closed expressions for this quantity for certain codebooks and when the codebook comprises all binary words, we show that this quantity is sub-quadratic when the deletion probability is less than 1/2. This then implies that the expected running time for this codebook is $o(M^3)$. For other codebooks, we develop methods to compute the expected number of edges efficiently. Finally, we adapt classical peeling-decoding techniques to reduce the number of nodes and edges in $\mathcal{G}_N$.

I. Introduction

In 1953, when Watson and Crick proposed the double helix model of the DNA molecule [1] , they wrote: "It has not escaped our notice that the specific pairing that we have postulated immediately suggests a possible copying mechanism for the genetic material." In the same year, the authors described the details of this replication mechanism [2] and more than seven decades later, the polymerase chain reaction (PCR) and other amplification techniques that exploit this copying mechanism have become an essential component in many bioengineering applications. Of interest to this paper are the applications in *DNA based data storage* (see [3], [4] for a survey) and *pooled testing of viral RNA* [5]–[9].

In both applications, to read the information on either a synthetic DNA data block or a viral RNA sample, the user typically employs a sequencing platform that creates multiple copies of the same strand. The sequencer then reads all these copies and provides multiple (possibly) errononeous reads to the user. Even though multiple reads allow the user to store more information or augment the testing capacity [10], [11], the *unsorted nature of DNA strands* poses certain computation problems. More concretely, in DNA based storage systems[1], a file is typically broken into many information blocks and stored onto different DNA strands, where their relative order is not preserved. Hence, when the user retrieves the information, in addition to decoding the data, the user has to determine the identity of the data that each strand stored. Now, a typical solution is to simply have a set of *addresses* and have each DNA strand to store this address information in its prefix. As the addresses are also known to the user, the user is able to identify the information after the decoding process.

[1]There are numerous works that address the unsorted nature of DNA-based data storage system. Due to space constraints, we provide only a short summary here and defer the detailed description in the extended version of this paper. Broadly, there are a few coding solutions: those that read all files [14]–[17], those that allow fast clustering [18], those that allow *random access* [19]–[21]. Also, fundamental limits of such storage systems are studied in [22]–[24].

However, as these addresses may also be corrupted, this solution requires further refinements and we discuss one experimental approach adopted by Organick *et al.* [12]. Here, the reads are first clustered with respect to the edit distance. Then the authors determine a consensus output amongst the reads in each cluster and finally, decode these consensus outputs using a classic concatenation scheme. For this approach, the clustering step is computationally expensive and in [13], a subset of the authors developed a distributed approximate clustering algorithm and clustered 5 billion reads in 46 minutes on 24 processors.

In this work, we study a method that avoids clustering. Here, we use the fact that the addresses $\mathcal{C}$ is available to the user. Instead of clustering the entire reads, we look at the collection of *prefixes* $\mathcal{Y}$ of the reads and assign each prefix $\boldsymbol{y} \in \mathcal{Y}$ to a certain address $\pi(\boldsymbol{y}) \triangleq \boldsymbol{x} \in \mathcal{C}$. If we assume certain channel characteristics, that is, the probability of prefix $\boldsymbol{y}$ given an address $\boldsymbol{x}$ is $P(\boldsymbol{y}|\boldsymbol{x})$, then the likelihood of an assignment can be computed to be $\prod_{\boldsymbol{y}\in\mathcal{Y}} P(\boldsymbol{y}|\pi(\boldsymbol{y}))$. Therefore, our optimization objective is to find an assignment that maximizes this probability. We formally define this problem in Section II.

We remark that our approach generalizes the bee identification problem originally proposed by Tandon *et al.* [25]. Informally, the *bee identification problem* requires the receiver to identify M "bees" using a set of M unordered noisy measurements. Tandon *et al.* studied the binary symmetric channel and showed that decoding the noisy measurements *jointly* results in a significantly smaller probability of erroneous identification [25]. Later, Kiah *et al.* investigated efficient ways of performing this joint decoding [26]. Specifically, for the binary erasure and binary symmetric channels, they reduced the bee-identification problem to certain combinatorial optimization problems. Then, applying well-known algorithms, they demonstrated that joint decoding can be performed in polynomial time (in M).

Here, we extend this model by assuming that each of the M bees results in N noisy measurements with $N \geq 1$, and we call this the *bee identification problem for multi-draw channels*. Our first contribution is to reduce this identification problem to the problem of finding a minimum-cost flow on a related flow network. Then, applying the Edmonds-Karp or Tomizawa algorithm [27], [28], we show that the bee identification problem for multi-draw channels can be solved in $O(M^3)$ time.

Similar to [26], our second contribution is a detailed study of this flow network in the context of *deletion channels*. Since the complexity of the network flow algorithm scales with the number of edges, we provide estimates on the expected number of edges. For certain codebooks, we obtain closed formulae for this quantity and in the case when $\mathcal{C} = \{0,1\}^n$, we show that the expected edge density of the network tends to zero when the deletion probability is less than $1/2$. This implies that the expected running time is sub-cubic. For other codebooks, determining the expected number of edges is challenging. Nevertheless, we develop techniques to compute this quantity in polynomial time (in n). Finally, we explore the use of *peeling decoders* to further reduce the number of nodes and edges in this flow network. In the next section, we formally define our problem and describe our contributions.

II. Problem Formulation

Let N and M be positive integers. Let $[M] \triangleq \{1,2,\ldots,M\}$. An N-permutation π over $[M]$ is an NM-tuple $(\pi(i))_{i\in[MN]}$ where every symbol in $[M]$ appears exactly N times, and we denote the set of all N-permutations over $[M]$ by $\mathbb{S}_N(M)$.

Let Σ be an alphabet of size two and we consider a length-n code $\mathcal{C} \subseteq \{0,1\}^n$ with M *codewords* $\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_M$. Consider, in addition, a channel $\mathcal{S}$ where the output $\boldsymbol{y}$ given an input $\boldsymbol{x}$ is received with probability $P(\boldsymbol{y}|\boldsymbol{x})$.

In our setup, we send *all* M codewords over the channel $\mathcal{S}$ and suppose that each codeword results in exactly N outputs. Therefore, we obtain an unordered multiset of MN outputs $\{\boldsymbol{y}_1,\boldsymbol{y}_2,\ldots,\boldsymbol{y}_{MN}\}$. Note that the outputs $\boldsymbol{y}_{iN-N+1},\boldsymbol{y}_{iN-N+2},\ldots,\boldsymbol{y}_{iN}$ are not necessarily the channel output of $\boldsymbol{x}_i$ and in fact, our task is to find an N-permutation π over $[M]$ such that $\boldsymbol{y}_i$ is most likely to be the channel output of the input $\boldsymbol{x}_{\pi(i)}$ for all $i \in [MN]$. Formally, assuming the channels are independent, our task is as follows.

(Bee Identification for Multi-draw Channels). To find an N-permutation π over $[M]$ so as to maximize the probability $\prod_{i=1}^{MN} P\left(\boldsymbol{y}_i|\boldsymbol{x}_{\pi(i)}\right)$.

A. A Flow Network

To perform our identification task, we define the following flow network $\mathcal{G}_N = (\mathcal{V},\mathcal{E},\boldsymbol{\gamma},\boldsymbol{\delta})$ using the M codewords $\mathcal{C} = \{x_i : i \in [M]\}$ and MN outputs $\mathcal{Y} = \{\boldsymbol{y}_j : j \in [MN]\}$.

(i) *Nodes*. The set of *left* and *right nodes* corresponds to the set of M codewords and the multiset of MN outputs, respectively. In other words, $\mathcal{V} \triangleq \mathcal{C} \cup \mathcal{Y}$.

(ii) *Demands*. For each left node / codeword $\boldsymbol{x} \in \mathcal{C}$, we assign the demand $\boldsymbol{\delta}(\boldsymbol{x}) \triangleq -N$, while for each right node / output $\boldsymbol{y} \in \mathcal{Y}$, we assign the demand $\boldsymbol{\delta}(\boldsymbol{y}) \triangleq 1$.

(iii) *Edges*. For a codeword $\boldsymbol{x}$ and an output $\boldsymbol{y}$, we draw the edge from $\boldsymbol{x}$ to $\boldsymbol{y}$ if and only if it is possible to obtain the channel output $\boldsymbol{y}$ from the input codeword $\boldsymbol{x}$, that is, $P(\boldsymbol{y}|\boldsymbol{x}) > 0$. Hence, $\mathcal{E} \triangleq \{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{C}\times\mathcal{Y} : P(\boldsymbol{y}|\boldsymbol{x}) > 0\}$.

(iv) *Costs*. For an edge $(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{C}\times\mathcal{Y}$, we assign the cost $\boldsymbol{\gamma}(\boldsymbol{x},\boldsymbol{y}) = -\log P(\boldsymbol{y}|\boldsymbol{x})$. Note that the cost is well-defined as the value $P(\boldsymbol{y}|\boldsymbol{x})$ is necessarily positive.

Given this flow network $\mathcal{G}_N$, the minimum-cost network flow problem is defined as follows.

$$\min \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{E}} f(\boldsymbol{x},\boldsymbol{y})\boldsymbol{\gamma}(\boldsymbol{x},\boldsymbol{y})$$

$$\text{such that} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{E}} f(\boldsymbol{x},\boldsymbol{y}) = -\boldsymbol{\delta}(\boldsymbol{x}) = N \qquad \text{for all } \boldsymbol{x}\in\mathcal{C}, \quad (1)$$

$$\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{E}} f(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{\delta}(\boldsymbol{y}) = 1 \qquad \text{for all } \boldsymbol{y}\in\mathcal{Y}, \quad (2)$$

$$f(\boldsymbol{x},\boldsymbol{y}) \in \{0,1\} \qquad \text{for all } (\boldsymbol{x},\boldsymbol{y})\in\mathcal{E}.$$

Consider a flow f in $\mathcal{G}_N$ with cost $\boldsymbol{\gamma}(f)$. That is, f fulfills both (1) and (2). We construct an N-permutation π as follows: set $\pi(j) = i$ if and only if $f(\boldsymbol{x}_i,\boldsymbol{y}_j) = 1$. It follows from (2) that $\pi(j)$ is assigned a value for all $j \in [MN]$. From (1), we have that every $i \in [M]$ appears exactly N times in π, and so, π is an N-permutation. Finally, we observe that $\prod_{i=1}^{MN} P\left(y_i|x_{\pi(i)}\right)$ is given by $2^{-\boldsymbol{\gamma}(f)}$. Therefore, minimizing the cost of a flow in $\mathcal{G}_N$ is equivalent to maximizing the probability for the bee-identification problem for multi-draw channels[2].

For the binary erasure (BEC) and binary symmetric channels (BSC), when $N = 1$, the preceding algorithm reduces to the bee-identification problem to the problem of finding a perfect matching and minimum-cost matching in $\mathcal{G}_1$, respectively [26]. In this work, we focus on the *deletion channel*, denoted by $\mathrm{Del}(p)$, where each bit in a sent codeword is independently deleted with probability p. More generally, the *deletion multi-draw channel*, denoted by $\mathrm{Del}(p;N)$, results in N independent outputs for each codeword sent through the deletion channel. Then for a codeword $\boldsymbol{x}$ and an output $\boldsymbol{y}$, the probability $P(\boldsymbol{y}|\boldsymbol{x})$ is given by $\mathrm{Emb}(\boldsymbol{x},\boldsymbol{y})p^d(1-p)^{n-d}$. Here, $d = |\boldsymbol{x}| - |\boldsymbol{y}|$ is the number of deletions, while $\mathrm{Emb}(\boldsymbol{x},\boldsymbol{y})$ denotes *embedding number of* $\boldsymbol{y}$ *in* $\boldsymbol{x}$, that is, the number of times $\boldsymbol{y}$ occurs as a subsequence of $\boldsymbol{x}$. When $P(\boldsymbol{y}|\boldsymbol{x}) > 0$, we draw an edge between $\boldsymbol{x}$ and $\boldsymbol{y}$ and we assign the cost $-\log\mathrm{Emb}(\boldsymbol{x},\boldsymbol{y}) - d\log p - (n-d)\log(1-p)$, where the logarithm base is two.

Example 1. Consider a multi-draw deletion channel with $p = 0.2$ and $N = 2$. Let $\mathcal{C}$ be the code $\{0000, 1001, 0110, 1111\}$ with $M = 4$ codewords of length four. This is a Varshamov-Tenengolts (VT) code [29].

Suppose that we pass all four words through the channel and obtain the following eight outputs are:

$$\boldsymbol{y}_1 = 0, \quad \boldsymbol{y}_2 = 11, \quad \boldsymbol{y}_3 = 11, \quad \boldsymbol{y}_4 = 000,$$
$$\boldsymbol{y}_5 = 000, \quad \boldsymbol{y}_6 = 0110, \quad \boldsymbol{y}_7 = 0110, \quad \boldsymbol{y}_8 = 1111.$$

Next, we describe the flow network $\mathcal{G}_N$. For the demand values, we simply have $\boldsymbol{\delta}(\boldsymbol{x}) = -2$ for $\boldsymbol{x} \in \mathcal{C}$ and $\boldsymbol{\delta}(\boldsymbol{y}) = 1$ for $\boldsymbol{y} \in \mathcal{Y}$. To display the cost values, we use a 4×8-table to reduce clutter. Here, the (i,j) entry is given by the cost of the edge $(\boldsymbol{x}_i,\boldsymbol{y}_j)$, i.e. $-\log P(\boldsymbol{x}_i,\boldsymbol{y}_j)$, and when there is no edge between $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$, we write the entry as ∞.

	$\boldsymbol{y}_1$	$\boldsymbol{y}_2$	$\boldsymbol{y}_3$	$\boldsymbol{y}_4$	$\boldsymbol{y}_5$	$\boldsymbol{y}_6$	$\boldsymbol{y}_7$	$\boldsymbol{y}_8$
$\boldsymbol{x}_1 = 0000$	5.288	∞	∞	1.288	1.288	∞	∞	∞
$\boldsymbol{x}_2 = 1001$	6.288	5.288	5.288	∞	∞	∞	∞	∞
$\boldsymbol{x}_3 = 0110$	6.288	5.288	5.288	∞	∞	1.288	1.288	∞
$\boldsymbol{x}_4 = 1111$	∞	2.703	2.703	∞	∞	∞	∞	1.288

Highlighted in blue are the edges whose flow values are one in a minimum-cost flow of $\mathcal{G}_N$. Notice that in each row and column, the number of blue edges are two and one, respectively. This meets the respective flow constraints (1) and (2). Then the corresponding maximum likelihood N-permutation is given by $(2,2,4,1,1,3,3,4)$. ■

In what follows, we discuss how to obtain this minimum-cost flow *efficiently*. To this end, we apply the algorithm of Edmonds and Karp [27], and Tomizawa [28], to compute a minimum-cost flow, and hence, a maximum likehood N-permutation, in $O(|\mathcal{V}|(|\mathcal{E}| + |\mathcal{V}|\log|\mathcal{V}|))$ time. However, in the worst case, the network $\mathcal{G}_N$ may be complete. That is, $|\mathcal{E}| = NM^2$ where M

[2]A reviewer pointed out that it is more reasonable to assume that each codeword results in *at most* N outputs. Now, we can modify the network in Section II-A to address this. Specifically, when there are $N' < MN$ right nodes, we include another $N' - MN$ right *auxiliary* nodes. For each of the M left nodes, we draw an edge to each right auxiliary node. The capacity and cost of each edge is then set to one and ∞, respectively. Then applying the minimum cost flow algorithm, we find the maximum-likehood N-permutation.

is the size of the code. Thus, in the worst case, the running time of this method is cubic in the number of codewords M.

However, observe that the network $\mathcal{G}_N$ in Example 1 is sparse, that is, $|\mathcal{E}| = 14$ is small as compared to $NM^2 = 32$. In this paper, under certain mild assumptions, we show that on average this is indeed the case, that is, the expected number of edges is $o(M^2)$. Specifically, for a codebook $\mathcal{C}$ and channel $\mathcal{S}$, we use $B_N(\mathcal{C}; \mathcal{S})$ to denote the expected number of edges in $\mathcal{G}_N$. When the codebook comprises all binary words of length n and the deletion probability p is less than $1/2$, we have the following result which is immediate from Proposition 11 in Section IV.

Theorem 1. *Let $\mathcal{S} = \mathrm{Del}(p; N)$ for fixed values of p and N. For $p < 1/2$, we have then $B_N(\{0,1\}^n; \mathcal{S}) \le NM^{1+\epsilon}$, where $M = 2^n$ and $0 < \epsilon < 1$ is a constant dependent only on p. Therefore, the expected running time of the minimum-cost flow algorithm is $o(M^3)$. Here, asymptotics are with respect to n.*

Furthermore, in Proposition 11, we show that the threshold $p = 1/2$ is tight. Specifically, the quantity $B_N(\{0,1\}^n)/NM^2$ tends to $1/2$ and 1, when $p = 1/2$ and $p > 1/2$, respectively.

To obtain this result, we provide closed formula for $B_1(\{0,1\}^n, \mathcal{S})$ using combinatorial techniques. Similar formulae are obtained for constant-weight and even-weight codebooks. For other codebooks, this enumeration problem is nontrivial and it is not clear that $B_N(\mathcal{C}; \mathcal{S})$ can be computed in polynomial time. Nevertheless, in the extended version, we use standard dynamic programming techniques to compute $B_N(\mathcal{C})$ in polynomial time for a class of codebooks. We remark that this class is rather general and includes many classical codes such as *linear codes* and *VT codes*.

B. Pruning with Peeling Decoder

Next, we reduce the running time of the network flow algorithm by pruning away certain nodes and edges. To do so, we modify the classic peeling decoders used in graph-based codes [35]. Intuitively, we search for degree-one nodes in the network $\mathcal{G}_N$. For any such node u with the edge uv, we must assign u to v. In such cases, we either remove u or v and the edge uv from the network. Specifically, we do the following.

- If the output $\boldsymbol{y}$ is a degree-one node and $\boldsymbol{x}$ is the only node adjacent to $\boldsymbol{y}$, we remove the output node $\boldsymbol{y}$ and the edge $\boldsymbol{xy}$. We also increase the demand of $\boldsymbol{x}$ by one and if the resulting demand is zero, we also remove the node $\boldsymbol{x}$ too.
- If the codeword $\boldsymbol{x}$ is a degree-one node and $\boldsymbol{y}$ is the only node adjacent to $\boldsymbol{x}$, we then remove both nodes $\boldsymbol{x}$ and $\boldsymbol{y}$ and all edges incident to the node $\boldsymbol{y}$.

We repeat this procedure until neither of these rules can be applied. That is, there is no degree-one node in the resulting flow network. We then denote this flow network by $\mathcal{G}_N^*$.

Example 2. Continuing Example 1, we remove nodes and edges from $\mathcal{G}_N$ according to the rules. Then the resulting network $\mathcal{G}_N^*$ has only codewords $\boldsymbol{x}_2$ and $\boldsymbol{x}_4$ with outputs $\boldsymbol{y}_2$ and $\boldsymbol{y}_3$.

	$\boldsymbol{y}_2$	$\boldsymbol{y}_3$
$\boldsymbol{x}_2 = 1001$	5.288	5.288
$\boldsymbol{x}_4 = 1111$	2.703	2.703

Here, the resulting demand is $\boldsymbol{\delta}(\boldsymbol{x}_2) = \boldsymbol{\delta}(\boldsymbol{x}_4) = -1$. Applying the network flow algorithm to $\mathcal{G}_N^*$ recovers the same solution as in Example 1. ■

As before, since the running time of the network flow algorithm depends on the number of nodes and edges, we are interested in determining the size of $\mathcal{G}_N^*$. Specifically, we estimate $A_N^*(\mathcal{C}; \mathcal{S})$ and $B_N^*(\mathcal{C}; \mathcal{S})$ which denote the expected number of nodes and edges, respectively, in $\mathcal{G}_N^*$.

III. Expected Number of Edges for General Multidraw Channels

Throughout this section, we fix some codebook $\mathcal{C}$ with M codewords. We send all M codewords through a general multidraw channel $\mathcal{S}$ and obtain NM noisy outputs. Following the preceeding section, we then construct the flow network $\mathcal{G}_N$. First, we study the expected number of edges in $\mathcal{G}_N$ for any general channel $\mathcal{S}$ and denote this quantity by $B_N(\mathcal{C}; \mathcal{S})$.

Now, let the codewords in $\mathcal{C}$ be $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$. For $i \in [M]$, let the N noisy outputs of $\boldsymbol{x}_i$ be $\widetilde{\boldsymbol{x}}_i^1, \widetilde{\boldsymbol{x}}_i^2, \ldots, \widetilde{\boldsymbol{x}}_i^N$. For $i, j \in [M]$ and $k \in [N]$, we consider the event that we insert an edge between codeword $\boldsymbol{x}_i$ and output $\widetilde{\boldsymbol{x}}_j^k$. Recall that we insert an edge if and only if the channel probability $P(\widetilde{\boldsymbol{x}}_j^k | \boldsymbol{x}_i)$ is strictly positive. As the expected number of edges in $\mathcal{G}_N$ is given by the sum of these probabilities, we have the following proposition.

Proposition 2. *If $Q(\boldsymbol{x} \prec \boldsymbol{x}')$ denote the probability that an output of $\boldsymbol{x}$ is also an output of $\boldsymbol{x}'$, then we have*

$$B_1(\mathcal{C}; \mathcal{S}) = \sum_{\boldsymbol{x} \in \mathcal{C}} \sum_{\boldsymbol{x}' \in \mathcal{C}} Q(\boldsymbol{x} \prec \boldsymbol{x}'), \tag{3}$$

$$B_N(\mathcal{C}; \mathcal{S}) = \sum_{k=1}^{N} \sum_{\boldsymbol{x} \in \mathcal{C}} \sum_{\boldsymbol{x}' \in \mathcal{C}} Q(\boldsymbol{x} \prec \boldsymbol{x}') = N B_1(\mathcal{C}; \mathcal{S}). \tag{4}$$

Therefore, it suffices to compute $B_1(\mathcal{C}; \mathcal{S})$ for general codebooks and channels. For the binary symmetric and binary erasure channels, we have the following results on the expected number of edges from [26].

Proposition 3 ([26])**.** *If $\mathcal{S}$ is the BSC, then $B_1(\mathcal{C}; \mathcal{S}) = |\mathcal{C}|^2$. If $\mathcal{S}$ is the BEC with erasure probability p, then $B_1(\mathcal{C}; \mathcal{S}) = D(p)$ where $D(z)$ is the* distance enumerator *of the code $\mathcal{C}$. Here, $D(z)$ is the polynomial $D(z) = \sum_{i=0}^{n} D_i z^i$, where D_i is the number of pairs of (not necessarily distinct) codewords with distance i.*

Since determining the distance enumerator $D(z)$ for a general linear code is NP-hard [36], we have that evaluating the quantity $B_1(\mathcal{C}; \mathcal{S})$ is also NP-hard[3] when $\mathcal{S}$ is the binary erasure channel. Therefore, we conjecture that the problem of determining $B_1(\mathcal{C}; \mathcal{S})$ is also difficult when $\mathcal{S}$ is the deletion channel. Nevertheless, in the next section, we study this problem and obtain closed formulas for certain special codebooks.

Here, we continue our discussion for general multidraw channels. Using the peeling decoder described in Section II-B, we can reduce the number of edges and vertices and obtain the flow network $\mathcal{G}_N^*$. Recall that $A_N^*(\mathcal{C}; \mathcal{S})$ and $B_N^*(\mathcal{C}; \mathcal{S})$ denote expected number of nodes and edges, respectively, in $\mathcal{G}_N^*$. It turns out we can bound these values using the quantity $B_1(\mathcal{C}; \mathcal{S})$ defined in (3).

[3]Suppose otherwise that there is a polynomial-time method to evaluate $D(p)$ for $0 \le p \le 1$. Then we can evaluate $D(z)$ at $n+1$ distinct points and recover the coefficients of $D(z)$ in polynomial time using Lagrange interpolation.

Similar to the case for $B_N(\mathcal{C};\mathcal{S})$, we observe that $A^*_N(\mathcal{C};\mathcal{S})$ and $B^*_N(\mathcal{C};\mathcal{S})$ depend on the case when $N=1$. Specifically, we have the following relations:

$$A^*_N(\mathcal{C};\mathcal{S}) \le N A^*_1(\mathcal{C};\mathcal{S}), \tag{5}$$

$$B^*_N(\mathcal{C};\mathcal{S}) = N B^*_1(\mathcal{C};\mathcal{S}). \tag{6}$$

Now, to provide upper bounds on both $A^*_1(\mathcal{C};\mathcal{S})$ and $B^*_1(\mathcal{C};\mathcal{S})$, we estimate the number of degree-one nodes in $\mathcal{G}_1$.

Proposition 4. *The expected number of degree-one left (codeword) nodes in $\mathcal{G}_1$ is at least $M - B_1(\mathcal{C};\mathcal{S})/2$.*

Proof. For any codeword $\boldsymbol{x}$, similar to the proof of Proposition 2, the expected degree of $\boldsymbol{x}$ (in $\mathcal{G}_1$) is $\sum_{\boldsymbol{x}'\in\mathcal{C}} Q(\boldsymbol{x}\prec\boldsymbol{x}')$, where $Q(\boldsymbol{x}\prec\boldsymbol{x}')$ is the probability that an output of $\boldsymbol{x}$ is also an output of $\boldsymbol{x}'$. Then by Markov inequality, the probability that $\boldsymbol{x}$ has degree at least two is at most $\frac{1}{2}\sum_{\boldsymbol{x}'\in\mathcal{C}} Q(\boldsymbol{x}\prec\boldsymbol{x}')$. That is the probability that $\boldsymbol{x}$ has degree one is at least $1-\frac{1}{2}\sum_{\boldsymbol{x}'\in\mathcal{C}} Q(\boldsymbol{x}\prec\boldsymbol{x}')$. Therefore, the expected number of degree-one left (codeword) nodes is at least

$$\sum_{x\in\mathcal{C}}\left(1-\frac{1}{2}\sum_{\boldsymbol{x}'\in\mathcal{C}} Q(\boldsymbol{x}\prec\boldsymbol{x}')\right) = M - \frac{B_1(\mathcal{C};\mathcal{S})}{2}. \qquad \square$$

Recall that all degree-one nodes and its corresponding edges must be removed at the first step. Hence, the following is immediate from Proposition 4.

Corollary 5.

$$A^*_1(\mathcal{C};\mathcal{S}) \le B_1(\mathcal{C};\mathcal{S}),\ \textit{and}\ B^*_1(\mathcal{C};\mathcal{S}) \le \frac{3B_1(\mathcal{C};\mathcal{S})}{2} - M.$$

IV. The Deletion Channel

Throughout this section, we have that $\mathcal{S} = \mathrm{Del}(p)$, for $0<p<1$. Observe from (4) and Corollary 5 that the quantity $B_1(\mathcal{C};\mathcal{S})$ is useful for providing estimates on sizes of the networks $\mathcal{G}_N$ and and $\mathcal{G}^*_N$ Hence, we study $B_1(\mathcal{C};\mathcal{S})$ and for brevity, write this quantity as $B(\mathcal{C})$. Our first proposition states that the problem of determining $B(\mathcal{C})$ is equivalent to certain enumeration problems concerning subsequences.

Proposition 6. *Fix some code $\mathcal{C}\subseteq\Sigma^n$. We have that*

$$B(\mathcal{C}) = \sum_{t=0}^{n}\sum_{\boldsymbol{z}\in\Sigma^{n-t}} I(\boldsymbol{z};\mathcal{C}) I^*(\boldsymbol{z};\mathcal{C}) p^t(1-p)^{n-t}. \tag{7}$$

Here, $I(\boldsymbol{z};\mathcal{C})$ denotes the number of words in $\mathcal{C}$ that contain $\boldsymbol{z}$ as a subsequence, while $I^*(\boldsymbol{z};\mathcal{C}) = \sum_{\boldsymbol{x}\in\mathcal{C}}\mathrm{Emb}(\boldsymbol{x},\boldsymbol{z})$. In other words, $I^*(\boldsymbol{z};\mathcal{C})$ counts the total number of occurrences of $\boldsymbol{z}$ amongst all codewords in $\mathcal{C}$.

Proof. Observe that for the $\mathrm{Del}(p)$-channel, the quantity $Q(\boldsymbol{x}\prec\boldsymbol{x}')$ denotes the probability that an output of $\boldsymbol{x}$ is a subsequence of $\boldsymbol{x}'$. So, for $0\le t\le n$, if we use $D_t(\boldsymbol{x})$ to denote the set of all $(n-t)$-subsequences of $\boldsymbol{x}$, then we have $Q(\boldsymbol{x}\prec\boldsymbol{x}') = \sum_{t=0}^{n}\sum_{z\in D_t(x)}\mathrm{Emb}(\boldsymbol{x},\boldsymbol{z})\mathbb{I}(\boldsymbol{z}\in\boldsymbol{x}')p^t(1-p)^{n-t}$. Here, we use $\mathbb{I}(\boldsymbol{z}\in\boldsymbol{x}')$ to denote the indicator function for the event that $\boldsymbol{z}$ is a subsequence of $\boldsymbol{x}'$. Using this expression and switching the order of summation, we can rewrite (3) as

$$\begin{aligned} &B(\mathcal{C}) \\ &= \sum_{t=0}^{n} p^t(1-p)^{n-t}\sum_{z\in\Sigma^{n-t}}\sum_{x\in\mathcal{C}}\sum_{x'\in\mathcal{C}}\mathrm{Emb}(\boldsymbol{x},\boldsymbol{z})\mathbb{I}(\boldsymbol{z}\in\boldsymbol{x}') \\ &= \sum_{t=0}^{n} p^t(1-p)^{n-t}\sum_{z\in\Sigma^{n-t}}\left(\sum_{x\in\mathcal{C}}\mathrm{Emb}(\boldsymbol{x},\boldsymbol{z})\right)\left(\sum_{x'\in\mathcal{C}}\mathbb{I}(\boldsymbol{z}\in\boldsymbol{x}')\right). \end{aligned}$$

Since $\sum_{\boldsymbol{x}\in\mathcal{C}}\mathrm{Emb}(\boldsymbol{x},\boldsymbol{z})$ and $\sum_{\boldsymbol{x}'\in\mathcal{C}}\mathbb{I}(\boldsymbol{z}\in\boldsymbol{x}')$ yields the quantities $I^*(\boldsymbol{z};\mathcal{C})$ and $I(\boldsymbol{z};\mathcal{C})$, respectively, we obtain (7). $\square$

Next, we look at the quantities $I(\boldsymbol{z};\mathcal{C})$ and $I^*(\boldsymbol{z};\mathcal{C})$ for the following codebooks:

$$\begin{aligned} \mathcal{A}_n &\triangleq \{0,1\}^n, \\ \mathcal{E}_n &\triangleq \{\boldsymbol{x}\in\{0,1\}^n : \mathrm{wt}(\boldsymbol{x}) \text{ is even}\}, \\ \mathcal{C}_{n,w} &\triangleq \{\boldsymbol{x}\in\{0,1\}^n : \mathrm{wt}(\boldsymbol{x}) = w\}. \end{aligned}$$

Now, the quantity $I(\boldsymbol{z};\mathcal{C})$ has been studied in other contexts [37], [38]. Of significance, $I(\boldsymbol{z};\{0,1\}^n)$ depends on $|\boldsymbol{z}|$ and n only, while $I(\boldsymbol{z};\mathcal{C}(n,w))$ depends on $|\boldsymbol{z}|$, $\mathrm{wt}(\boldsymbol{z})$, n, and w only. In both cases, the quantity does not depend on the actual string $\boldsymbol{z}$. Specifically, we have the following proposition.

Proposition 7 ([37], [38])**.** *Let $|\boldsymbol{z}| = n-t$. Then*

$$I(\boldsymbol{z};\mathcal{A}_n) = \sum_{i=0}^{t}\binom{n}{i} \triangleq I_A(n,t). \tag{8}$$

Furthermore, if $\mathrm{wt}(\boldsymbol{z}) = u$,

$$\begin{aligned} &I(\boldsymbol{z};\mathcal{C}_{n,w}) \\ &= \begin{cases} \binom{n}{w}, & \textit{if } u=0 \textit{ and } w\le t, \\ \sum_{i=u}^{t-w+2u}\binom{i-1}{u-1}\binom{n-i}{w-u}, & \textit{if } u>1 \end{cases} \\ &\triangleq I_C(n,w,t,u). \end{aligned} \tag{9}$$

Next, using standard techniques in combinatorics [39], we have the following result. If $|\boldsymbol{z}| = n-t$ and $\mathrm{wt}(\boldsymbol{z}) = u$, then

$$I^*(\boldsymbol{z};\mathcal{A}_n) = 2^t\binom{n}{t}, \tag{10}$$

$$I^*(\boldsymbol{z};\mathcal{C}_{n,w}) = \binom{n}{t}\binom{t}{w-u}, \tag{11}$$

$$I^*(\boldsymbol{z};\mathcal{E}_n) = \begin{cases} 2^{t-1}\binom{n}{t}, & \text{if } t\ge 1, \\ 1, & \text{if } t=0 \text{ and } u \text{ is even}, \\ 0, & \text{if } t=0 \text{ and } u \text{ is odd}. \end{cases} \tag{12}$$

Using (7–11), we then obtain the following closed formulae for the expected number of edges. We defer the detailed proof to the extended version of this paper.

Theorem 8. *For all n,*

$$B(\mathcal{A}_n) = 2^n\sum_{t=0}^{n}\sum_{i=0}^{t}\binom{n}{i}\binom{n}{t}p^t(1-p)^{n-t}. \tag{13}$$

For $w\le n$,

$$\begin{aligned} B(\mathcal{C}_{n,w}) = \sum_{t=0}^{n}\sum_{u=0}^{w}&\binom{n-t}{u}\binom{n}{t}\binom{t}{w-u} \\ &\cdot I_C(n,t,w,u)p^t(1-p)^{n-t}. \end{aligned} \tag{14}$$

Now, for the codebook with even-weight words, one can apply (9) and (12) directly. However, the formula becomes unwieldy. Nevertheless, in what follows, we show that the expected number of edges when $\mathcal{C} = \mathcal{E}_n$ is approximately a quarter of the expected number of edges when $\mathcal{C} = \mathcal{A}_n$. Specifically, we have the following theorem.

Proposition 9. *If n is odd,*

$$B(\mathcal{E}_n) = \frac{1}{4}B(\mathcal{A}_n) + 2^{n-2}(1-p)^n. \qquad (15)$$

To prove this proposition, we use the following combinatorial lemma whose proof is deferred to the journal version.

Lemma 10. *Let $0 \le t \le n$. If n is odd, then*

$$\sum_{\boldsymbol{z}\in\{0,1\}^{n-t}} I(\boldsymbol{z};\mathcal{E}_n) = \frac{1}{2}\sum_{\boldsymbol{z}\in\{0,1\}^{n-t}} I(\boldsymbol{z};\mathcal{A}_n). \qquad (16)$$

Proof of Proposition 9. It is immediate from (10), (12) and (16) that for $t \ge 1$,

$$\sum_{\boldsymbol{z}\in\{0,1\}^{n-t}} I(\boldsymbol{z};\mathcal{E}_n)I^*(\boldsymbol{z};\mathcal{E}_n) = \frac{1}{4}\sum_{\boldsymbol{z}\in\{0,1\}^{n-t}} I(\boldsymbol{z};\mathcal{A}_n)I^*(\boldsymbol{z};\mathcal{A}_n)\,.$$

So, we have that

$$\begin{aligned} B(\mathcal{E}_n) &= \frac{1}{4}\left(B(\mathcal{A}_n) - 2^n(1-p)^n\right) + 2^{n-1}(1-p)^n \\ &= \frac{1}{4}B(\mathcal{A}_n) + 2^{n-2}(1-p)^n. \end{aligned}$$ □

A. Polarization of Edge Density

In this subsection, we consider a family of codebooks $\{\mathcal{C}_n : n \ge 1\}$ with increasing block lengths n, and study the quantity $\lim_{n\to\infty} B(\mathcal{C}_n)/|\mathcal{C}_n|^2$. Observe that $B(\mathcal{C}_n)/|\mathcal{C}_n|^2$ represents the expected edge density of the graph $\mathcal{G}_1$. When $\mathcal{C}_n = \mathcal{A}_n$, the next proposition states that this density approaches zero whenever the deletion probability is strictly less than half.

Proposition 11. *Let $\mathcal{C} = \mathcal{A}_n$ and $M = 2^n$. If $0 \le p < 1/2$, then $B_1(\mathcal{C}) \le M^{1+\epsilon}$ for some constant $0 < \epsilon < 1$ that is dependent only on p.*

Sketch of proof. The detailed argument is deferred to the full version of the paper. Instead, we simply provide the choice of ϵ for which the upper bound is valid. We choose ϵ such that $\epsilon \ge \max\{H(p), H(\beta) + 2(\log n)/n\} + 1/n$. Here, $\beta = \alpha/(1+\alpha)$ with $\alpha = \sqrt{p/(1-p)}$. □

Applying (4), we obtain Theorem 1 and hence, on average, the running time of the network flow algorithm is sub-cubic. Next, to complete our analysis, we show that the threshold $p = 1/2$ is tight and that the edge density polarizes. That is, when $p = 1/2$, we no longer have the property that $B(\mathcal{A}_n) = o(M^2)$ and when $p > 1/2$, we have $B(\mathcal{A}_n)$ approaches M^2. Before we formally state the result, we observe that $B(\mathcal{E}_n)$ shares the same polarization behavior as $B(\mathcal{A}_n)$. This follows directly from (15).

Proposition 12.

$$\lim_{n\to\infty}\frac{B(\mathcal{E}_n)}{|\mathcal{E}_n|^2} = \lim_{n\to\infty}\frac{B(\mathcal{A}_n)}{|\mathcal{A}_n|^2} = \begin{cases} 0, & \text{if } p < 1/2\ , \\ \frac{1}{2}, & \text{if } p = 1/2\ , \\ 1, & \text{if } p > 1/2\ . \end{cases}$$

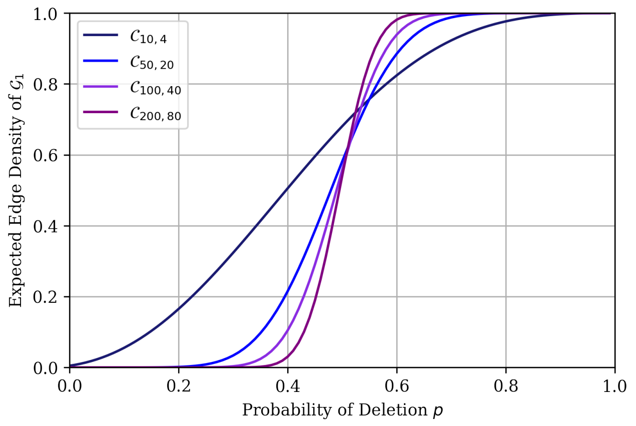

Fig. 1. Expected edge density of $\mathcal{G}_1$ for constant weight codebooks $C_{n,\omega n}$ with $\omega = 0.4$.

We defer the proof to the full version. Here, we conjecture that the same polarization phenomenon is present for constant-weight codebooks.

Conjecture. Let $0 < \omega < 1/2$. There exists $0 < p_\omega < 1$, a constant dependent on ω only, such that

$$= \lim_{n\to\infty}\frac{B(\mathcal{C}_{n,\lfloor\omega n\rfloor})}{|\mathcal{C}_{n,\lfloor\omega n\rfloor}|^2} = \begin{cases} 0, & \text{if } p < p_\omega\ , \\ 1, & \text{if } p > p_\omega\ . \end{cases}$$

We exhibit this polarization phenomenon numerically for the case $\omega = 0.4$ in Figure 1.

B. Enumeration via Dynamic Programming

Here, we consider the problem of enumerating $\mathcal{B}_1(\mathcal{C})$ for a certain class of codebooks. Consider a finite ring R that contains the q-ary alphabet Σ. We fix $\boldsymbol{H}$ to be a $r \times n$-matrix over R and $\boldsymbol{\sigma}$ be some syndrome in R^r. Then we consider the codebook $\mathcal{C}$ which is defined to be

$$\mathcal{C} \triangleq \{\boldsymbol{x} \in \Sigma^n : \boldsymbol{x}\boldsymbol{H}^T = \boldsymbol{\sigma}\}\,.$$

If a codebook satisfies this definition, we say that the codebook is *defined by a linear syndrome* and we observe that this general definition includes many classical codes.

- If $R = \mathbb{Z}_{n+1}$, $\Sigma = \{0,1\}$, $\boldsymbol{H} = (1,2,\ldots,n)$ and $\boldsymbol{\sigma} = (a)$ where $a \in \mathbb{Z}_{n+1}$, then the resuling codebook $\mathcal{C}$ is the Varshamov-Tenengolts code [29].
- If $R = \Sigma = \mathbb{F}_2$, $\boldsymbol{H} = (1,1,\ldots,1)$ and $\boldsymbol{\sigma} = (0)$. Then $\mathcal{C} = \mathcal{E}_n$. More generally, if we allow $\boldsymbol{H}$ to be any parity-check matrix, then $\mathcal{C}$ is a binary linear code.

Due to space constraints, we do not describe the detailed recursive formulas and the dynamic programming implementation. Instead, we simply state the running time of our proposed enumeration method.

Proposition 13. *Let $\mathcal{C}$ be a q-ary code defined by a linear syndrome with ring R, $r \times n$-matrix $\mathbf{H}$ and syndrome $\boldsymbol{\sigma}$. Then $B(\mathcal{C})$ can be determined in $O(q^2 n^4 |R|^{2r})$ time.*

In particular, if $\mathcal{C}$ is a VT code of length n, we can determine $B(\mathcal{C})$ in $O(n^6)$ time.

Acknowledgement

This research of Han Mao Kiah is supported by the Ministry of Education, Singapore, under its MOE AcRF Tier 2 Award MOE-T2EP20121-0007. The research of Eitan Yaakobi is supported by the Israel Innovation Authority grant 75855.

References

[1] J. D. Watson and F. H. Crick, "Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid," Nature, 171(4356), pp. 737-738, 1953.

[2] J. D. Watson and F. H. Crick, "The structure of DNA," In Cold Spring Harbor symposia on quantitative biology, Vol. 18, pp. 123-131, Jan. 1953. Cold Spring Harbor Laboratory Press.

[3] S. Yazdi, H. M. Kiah, E. R. Garcia, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: Trends and methods." *IEEE Trans. Molecular, Biological, Multi-Scale Commun.*, vol. 1, no. 3, pp. 230-248, 2015.

[4] O. Milenkovic, R. Gabrys, H. M. Kiah, and S. H. T. Yazdi, "Exabytes in a test tube," *IEEE Spectrum*, vol. 55, no. 5, pp. 40–45, 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] V.I. Levenshtein, "Efficient reconstruction of sequences," *IEEE Trans. on Inform. Theory*, vol. 47, no. 1, pp. 2–22, Jan. 2001.

[11] K. Cai, H. M. Kiah, T. T. Nguyen and E. Yaakobi, "Coding for Sequence Reconstruction for Single Edits," *IEEE Transactions on Information Theory*, vol. 68, no. 1, pp. 66-79, Jan. 2022.

[12] L. Organick, S. Ang, Y.-J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M. Racz, G. Kamath, P. Gopalan, B. Nguyen, C. Takahashi, S. Newman, H.-Y. Parker, C. Rashtchian, K. Stewart, G. Gupta, R. Carlson, J. Mulligan, D. Carmean, G. Seelig, L. Ceze, and K. Strauss, "Random access in large-scale DNA data storage", Nature Biotechnology, vol. 36, no. 3, pp 242-248, 2018.

[13] C. Rashtchian, K. Makarychev, M. Racz, S. Ang, D. Jevdjic, S. Yekhanin, L. Ceze, and K. Strauss, "Clustering billions of reads for DNA data storage," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[14] M. Kovacević and V. Y. F. Tan, "Codes in the space of multisets coding for permutation channels with impairments," *IEEE Trans. on Inform. Theory*, vol. 64, no. 7, pp. 5156–5169, Jul. 2018.

[15] A. Lenz, P. H. Siegel, A. Wachter-Zeh and E. Yaakobi, "Coding Over Sets for DNA Storage," *IEEE Transactions on Information Theory*, vol. 66, no. 4, pp. 2331-2351, April 2020.

[16] J. Sima, N. Raviv and J. Bruck, "On Coding Over Sliced Information," *IEEE Transactions on Information Theory*, vol. 67, no. 5, pp. 2793-2807, May 2021.

[17] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Anchor-based correction of substitutions in indexed sets," *Proc. IEEE Int. Symp. Inf. Theory*, Paris, France, pp. 757–761, Jul. 2019.

[18] T. Shinkar, E. Yaakobi, A. Lenz and A. Wachter-Zeh, "Clustering-Correcting Codes," *IEEE Transactions on Information Theory*, doi: 10.1109/TIT.2021.3127174.

[19] S. Yazdi, H. M. Kiah, R. Gabrys, and O. Milenkovic. Mutually uncorrelated primers for DNA-based data storage. *IEEE Transactions on Information Theory*, vol. 64, no. 9, pp. 6283-6296, 2018.

[20] M. Levy and E. Yaakobi, "Mutually uncorrelated codes for DNA storage," *IEEE Transactions on Information Theory*, vol. 65, no. 6, pp. 3671–3691, 2018.

[21] Y. M. Chee, H. M. Kiah and H. Wei, "Efficient and Explicit Balanced Primer Codes," *IEEE Transactions on Information Theory*, vol. 66, no. 9, pp. 5344-5357, Sept. 2020.

[22] R. Heckel, I. Shomorony, K. Ramchandran and D. N. C. Tse, "Fundamental limits of DNA storage systems," *Proc. IEEE Int. Symp. on Inform. Theory*, pp. 3130–3134, Aachen, Germany, Jun. 2017.

[23] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "An upper bound on the capacity of the DNA storage channel," *Proc. IEEE Inf. Theory Workshop*, pp. 1–5, Frisby, Sweden, Aug. 2019.

[24] I. Shomrony and R. Heckel, "Capacity results for the noisy shuffling channel," *Proc. IEEE Int. Symp. Inf. Theory*, Paris, France, pp. 762–766, Jul. 2019.

[25] 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.

[26] H. M. Kiah, A. Vardy and H. Yao, "Efficient Bee Identification," *Proc. IEEE Int. Symp. Inf. Theory*, Melbourne, Australia, pp. 1943-1948, 2021.

[27] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," Journal of the ACM (JACM), vol. 19, no. 2, pp. 248-264, 1972.

[28] N. Tomizawa, "On some techniques useful for solution of transportation network problems." *Networks*, vol. 1, no. 2, pp. 173-194, 1971.

[29] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals" (in Russian), Doklady Akademii Nauk SSSR, vol. 163, no. 4, 1965. English translation in Soviet Physics Dokl., vol. 10, no. 8, 1966.

[30] G. M. Church, Y. Gao, and S. Kosuri. "Next-generation digital information storage in DNA," *Science*, vol. 337, no. 6102, pp. 1628–1628, 2012.

[31] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney. "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 7435, pp. 77–80, 2013.

[32] M. Blawat, K. Gaedke, I. Hütter, X.-M. Chen, B. Turczyk, S. Inverso, B.W. Pruitt, and G.M. Church, "Forward error correction for DNA data storage," *Int. Conf. on Computational Science*, vol. 80, pp. 1011–1022, 2016.

[33] Y. Erlich and D. Zielinski, "DNA Fountain enables a robust and efficient storage architecture." *Science*, vol. 355, no. 6328, pp. 950-954, 2017.

[34] S. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Scientific reports*, vol. 7, no. 1, pp. 5011, 2017.

[35] V. V. Zyablov and M. S. Pinsker, "Estimation of the error-correction complexity of Gallager low-density codes," *Problems of Information Transmission*, vol. 11, no. 1, pp. 18–28, 1976.

[36] A. Vardy, "Algorithmic complexity in coding theory and the minimum distance problem." In Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, pp. 92-109, May 1997.

[37] V. I. Levenshtein, "Elements of coding theory." Diskretnaya matematika i matematicheskie voprosy kibernetiki, pp. 207-305, 1974.

[38] A. Atashpendar, M. Beunardeau, A. Connolly, R. Géraud, D. Mestel, A. W. Roscoe, and P. Y. Ryan, "From clustering supersequences to entropy minimizing subsequences for single and double deletions," 2018. arXiv preprint arXiv:1802.00703.

[39] P. Flajolet and R. Sedgewick (2009). Analytic combinatorics. Cambridge University press.