Bee Identification Problem for DNA Strands

Johan Chrisnata, Han Mao Kiah, *Senior Member, IEEE*, Alexander Vardy, and Eitan Yaakobi, *Senior Member, IEEE*

***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 this unordered set of MN noisy outputs. First, via a reduction to a minimum-cost flow problem on a related bipartite flow network called the *input-output flow network*, we show that the problem can be solved in $O(M^3)$ time in the worst case. Next, we consider the deletion and the insertion channels individually, and in both cases, we study the expected number of edges in their respective input-output networks. 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 or insertion probability is less than 1/2. This then implies that the expected running time to perform joint decoding 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 the input-output flow network.**

***Index Terms*—Bee identification problem, DNA-based data storage, multidraw channels, deletion channels.**

I. Introduction

In 1953, when Watson and Crick proposed the double helix model of the DNA molecule [2], 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 [3] 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 following applications.

1) *DNA based data storage:* Here, digital information is written onto synthetic DNA strands, that are in turn stored in a container in an unordered manner. Since the first experiments conducted by Church et al. in 2012 [4] and Goldman et al. in 2013 [5], there have been a flurry of experimental demonstrations (see [6], [7], [8] for a survey). To date, the "largest" experiment is due to Organick et al. where the amount of data stored is 200MB [9].
2) *Pooled Testing of Viral RNA:* Recently, to increase the testing throughput for COVID-19 infections, Schmid-Burgk et al. developed a procedure where multiple DNA samples are pooled, sequenced and analyzed *en masse* [10]. Unlike classical group testing, Schmid-Burgk et al. inserted barcodes / codewords in each sample to facilitate identification. As before, during this testing procedure, multiple copies of each DNA strand are created and the authors were able to reliably identify the viral samples. Later, similar experiments were replicated with different codebook / barcode spaces demonstrating the feasibility of the pooled testing approach [11], [12], [13], [14].

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 [15], [16], the *unsorted nature of DNA strands* poses certain computation problems. More concretely, in DNA based storage system,[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.

Manuscript received 12 January 2023; revised 24 April 2023 and 1 June 2023; accepted 28 June 2023. Date of publication 11 July 2023; date of current version 3 August 2023. The work of Han Mao Kiah was supported by the Ministry of Education, Singapore, under its MOE AcRF Tier 2 Award under Grant MOE-T2EP20121-0007. The work of Alexander Vardy was supported in part by the National Science Foundation (NSF) under Grant CCF-2212437. The work of Eitan Yaakobi was supported in part by the National Science Foundation (NSF) under Grant CCF-2212437, and in part by the European Union (ERC, DNAStorage) under Grant 865630. Parts of this work were presented in the IEEE International Symposium on Information Theory (ISIT2022) [DOI: 10.1109/ISIT50566.2022.9834414]. *(Corresponding author: Han Mao Kiah.)*

Johan Chrisnata is with the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore (e-mail: johan.chrisnata@ntu.edu.sg).

Han Mao Kiah is with the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore (e-mail: hmkiah@ntu.edu.sg).

Alexander Vardy, deceased, was with the Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093 USA (e-mail: avardy@ucsd.edu).

Eitan Yaakobi is with the Department of Computer Science, Technion—Israel Institute of Technology, Haifa 32000, Israel (e-mail: yaakobi@cs.technion.ac.il).

Digital Object Identifier 10.1109/JSAIT.2023.3294423

[1]There are numerous works that address the unsorted nature of DNA-based data storage system and we provide a short survey at the end of this introduction.

However, as these addresses may also be corrupted, this solution requires further refinements and we discuss one experimental approach adopted by Organick et al. [9]. 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 [17], 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 codebook $\mathcal{C}$ is available to the user. Instead of clustering the entire reads, we look at the collection[2] 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. [18]. 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 [18]. Later, Kiah et al. investigated efficient ways of performing this joint decoding [19]. 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 bipartite flow network, which we call the *input-output flow network*. Then, applying the Edmonds-Karp or Tomizawa algorithm [21], [22], we show that the bee identification problem for multi-draw channels can be solved in $O(M^3)$ time, where N is fixed. To reduce the running time complexity, we explore the use of *peeling decoders* to further reduce the number of nodes and edges in the input-output flow network in Section II-D.

Since the complexity of the network flow algorithm scales with the number of edges, we provide estimates on the expected number of edges. In Section III, we first study this number for any general channel $\mathcal{S}$. Next, similar to [19], our second contribution is a detailed study of the input-output flow network in the context of *deletion channels* in Section IV and *insertion channels* in Section V. Since the analysis for the deletion channels and for the insertion channels are similar, we focus our in-depth analysis on the deletion channels. For certain codebooks, we obtain closed formulae for the expected number of edges 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 of the algorithm for the deletion channel 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) for any code that can be defined with *linear syndrome* in Section IV-B. In the next section, we formally define our problem and describe our contributions.

[2]As pointed out a reviewer, here we are assuming that we are able to accurately determine the corrupted prefix. A naive, and slightly costly, solution is to separate the index and the file in each strand via some marker sequence, like a run of ℓ zeroes, and then forbid the file from containing such a run. A comprehensive study is given by [20]. A potential research direction is to determine the optimality of this solution.

A. DNA-Based Data Storage

For completeness, we survey some works that address the unsorted nature of DNA-based data storage system.

1) *Clustering-Correcting Codes:* As described earlier, to protect against the corruption, a solution is to cluster the reads with respect to certain metric [9], [17]. As this approach is computationally expensive, the authors in [23] proposed a new family of codes called *clustering-correcting codes:* These codes ensure that if the distance between the addresses of two strands is small, then the distance between their data blocks is large. In [23], the authors then exploited this property to cluster the strands correctly, even in the presence of errors.
2) *Coding over Sets:* To study this storage systems, another line of work proposed a new channel model where data is sent as an unordered set of strings. The channel is sometimes referred to as the *shuffling channel* (see [8] and the references therein), while the code design problem is referred to as *coding over sets* [24]. Families of such codes are constructed in [24], [25], [26], while fundamental limits of such channels are studied in [8], [27], [28], [29], [30].
3) *Coding for Random Access:* In the previous approach, all files have to be read to retrieve any information. In order to read a specific block of the information, Yazdi et al. proposed a strategy that exploits the DNA hybridization process to randomly access encoded DNA strands. Coding design considerations were provided in [31], while explicit codes were constructed in [32], [33].

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 Σ^n denote the set of all binary words of length n. Let $\Sigma^* = \cup_{n=0}^{\infty} \Sigma^n$. We consider a length-$n$ code $\mathcal{C} \subseteq \Sigma^n$ 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 $\prod_{i=1}^{MN} P(\boldsymbol{y}_i|\boldsymbol{x}_{\pi(i)})$.

We emphasize that codebook $\mathcal{C}$ is known to both the sender and receiver. In particular, the channel inputs $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$ are known to the receiver and the receiver's task is to assign each channel output $\boldsymbol{y}$ to some channel input $\boldsymbol{x}$.

A. A Bipartite Flow Network

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

1) *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, $\mathcal{V} \triangleq \mathcal{C} \cup \mathcal{Y}$.
2) *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 $\boldsymbol{\delta}(\boldsymbol{y}) \triangleq 1$.
3) *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, $\mathcal{E} \triangleq \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{C} \times \mathcal{Y} : P(\boldsymbol{y}|\boldsymbol{x}) > 0\}$.
4) *Costs:* For an edge $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{C} \times \mathcal{Y}$, we assign the cost $\boldsymbol{\gamma}(\boldsymbol{x}, \boldsymbol{y}) = -\log_2 P(\boldsymbol{y}|\boldsymbol{x})$. Note that the cost is well-defined as the value $P(\boldsymbol{y}|\boldsymbol{x})$ is necessarily positive.

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

$$
\begin{aligned}
\min \quad & \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{E}} f(\boldsymbol{x}, \boldsymbol{y})\boldsymbol{\gamma}(\boldsymbol{x}, \boldsymbol{y}) \\
\text{s.t.} \quad & \sum_{\boldsymbol{y}\in\mathcal{Y}} f(\boldsymbol{x}, \boldsymbol{y}) = -\boldsymbol{\delta}(\boldsymbol{x}) = N \text{ for every } \boldsymbol{x} \in \mathcal{C}, \qquad (1) \\
& \sum_{\boldsymbol{x}\in\mathcal{C}} f(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\delta}(\boldsymbol{y}) = 1 \text{ for every } \boldsymbol{y} \in \mathcal{Y}, \qquad (2) \\
& f(\boldsymbol{x}, \boldsymbol{y}) \in \{0, 1\} \quad \text{for all } (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{E}.
\end{aligned}
$$

Consider a flow f in $\mathcal{G}_N$ with cost $\boldsymbol{\gamma}(f) \triangleq \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{E}} f(\boldsymbol{x}, \boldsymbol{y})\boldsymbol{\gamma}(\boldsymbol{x}, \boldsymbol{y})$. 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 $\prod_{i=1}^{MN} P(\boldsymbol{y}_i|\boldsymbol{x}_{\pi(i)})$ 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.[3]

For the binary erasure channel (BEC) and binary symmetric channel (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 [19]. In this work, we focus on the *deletion channel* and the *insertion channel*, denoted by Del(p) and Ins(p) respectively.

We provide a similar definition of the probabilistic deletion and insertion channels as in [34] with little modification, described in a general setting as follows.

Definition 1: The general insertion/deletion channel sequentially takes an input symbol from a sent codeword $\boldsymbol{x} = x_1x_2, \ldots x_n$ of length n, and constructs a variable length output $\boldsymbol{y} = y_1y_2 \cdots \in \Sigma^*$ sequentially. The general channel is defined by three parameters, namely p_{ins}, p_{del}, and p_{cor}, where $p_{ins} + p_{del} + p_{cor} = 1$. Let the input pointer be i and the output pointer be j. The initial pointer positions are at $i = 1$ and $j = 1$. For the purpose of this paper, we assume that $x_{n+1} = \epsilon$ is an empty bit. This is to allow the possibility of more insertions happening after the last bit x_n. Iteratively, we sample one of the following three events with their corresponding probabilities until $i = n + 2$.

- *Insertion with probability p_{ins}.* Choose y_j uniformly at random from Σ. Increase j by one.
- *Deletion with probability p_{del}.* Increase i by one.
- *Correct with probability p_{cor}.* Set $y_j = x_i$. Increase both i and j by one.

In [34], the authors included an additional parameter p_{sub} to denote the substitution probability. For this paper, we focus on the pure *deletion channel* (without insertion) and pure *insertion channel* (without deletion). One future direction is to consider combinations of deletions and insertions occurring in the same channel. In the *deletion channel*, we set p_{del} to be a positive number $0 < p < 1$ and p_{ins} to be zero. In other words, each bit in a sent codeword is independently deleted with probability $p > 0$. More generally, the *deletion multi-draw channel*, denoted by Del$(p; N)$, results in N independent outputs for each codeword sent through the deletion channel. Similarly, for the *insertion channel*, we set p_{del} to be zero and p_{ins} to be a positive number $0 < p < 1$. More generally, the *insertion multi-draw channel*, denoted by Ins$(p; N)$, results in N independent outputs for each codeword sent through the insertion channel. In the next few subsections, we discuss in detail how we define the edge costs corresponding to the deletion and insertion channels.

B. The Deletion Channel

First, we describe the solution and our result for the deletion channel, where the probability p_{del} is a positive number $0 <$

[3]This problem can be generalized to the case when each codeword results in *at most* N outputs. In this case, we can modify the network in Section II-A to address this. Specifically, when there are $N' < MN$ right nodes, we include another $MN - N'$ right *auxiliary* nodes. For each of the M left nodes, we draw an edge to each right auxiliary node. The cost of each edge is then set to ∞. Then applying the minimum cost flow algorithm, we find the maximum-likehood N-permutation.

$p < 1$. 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 $-\log_2 \mathrm{Emb}(\boldsymbol{x},\boldsymbol{y}) - d\log_2 p - (n-d)\log_2(1-p)$, where the logarithm base is two.

Example 1: Consider the multi-draw deletion channel with $p = 0.2$ and $N = 2$. Let $\mathcal{C} = \{0000, 1001, 0110, 1111\}$ be the code with $M = 4$ codewords of length four. This is a Varshamov-Tenengolts (VT) code [35].

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

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

Next, we describe the input-output 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., $-\log_2 P(\boldsymbol{x}_i,\boldsymbol{y}_j)$, 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 $(2, 2, 4, 1, 1, 3, 3, 4)$. Note that a minimum-cost flow, or equivalently, a maximum-likelihood N-permutation, is not unique. We refer to the reader to the concluding remarks for a discussion. ■

In what follows, we discuss how to obtain this minimum-cost flow *efficiently*. To this end, we apply the algorithm of Edmonds and Karp [21], and Tomizawa [22], to compute a minimum-cost flow, and hence, a maximum likehood N-permutation, in $O(|\mathcal{V}|(|\mathcal{E}|+|\mathcal{V}|\log_2|\mathcal{V}|))$ time. However, in the worst case, the network $\mathcal{G}_N$ may form a complete bipartite graph. That is, $|\mathcal{E}| = NM^2$ where M 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 input-output 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 the input-output flow network $\mathcal{G}_N$. 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 8 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 8, we show that the threshold $p = 1/2$ is tight. Specifically, we demonstrate that the quantity $B_N(\{0,1\}^n;\mathcal{S})/NM^2$ 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 Section IV-B, we use standard dynamic programming techniques to compute $B_N(\mathcal{C};\mathcal{S})$ 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*.

C. The Insertion Channel

Similar to the deletion channel, we now describe the solution and our result for the insertion channel, where the probability p_{ins} is a positive number $0 < p < 1$. For a codeword $\boldsymbol{x}$ and an output $\boldsymbol{y}$, the probability $P(\boldsymbol{y}|\boldsymbol{x})$ is given by $\mathrm{Emb}(\boldsymbol{y},\boldsymbol{x})p^d(1-p)^n$, where $d = |\boldsymbol{y}| - |\boldsymbol{x}|$ is the number of insertions. When $P(\boldsymbol{y}|\boldsymbol{x}) > 0$, we draw an edge between $\boldsymbol{x}$ and $\boldsymbol{y}$ and we assign the cost $-\log_2 \mathrm{Emb}(\boldsymbol{y},\boldsymbol{x}) - d\log_2 p - n\log_2(1-p)$, where the logarithm base is two. Similarly to the deletion channel, we then apply the algorithm of Edmonds and Karp [21], and Tomizawa [22], to compute a minimum-cost flow in $O(|\mathcal{V}|(|\mathcal{E}|+|\mathcal{V}|\log_2|\mathcal{V}|))$ time.

D. Pruning With the 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 [36]. Intuitively, we search for degree-one nodes in the input-output 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 $\boldsymbol{y}_3$.

	y_2	y_3
$x_2 = 1001$	5.288	5.288
$x_4 = 1111$	2.703	2.703

Here, the resulting demand is $\boldsymbol{\delta}(x_2) = \boldsymbol{\delta}(x_4) = -1$. 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 input-output 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 random noisy outputs of $\boldsymbol{x}_i$ be $\boldsymbol{y}_{i,1}, \boldsymbol{y}_{i,2}, \ldots, \boldsymbol{y}_{i,N}$. 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 $\boldsymbol{y}_{j,k}$. Recall that we insert an edge if and only if the channel probability $P(\boldsymbol{y}_{j,k}|\boldsymbol{x}_i)$ 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 1: If $Q(\boldsymbol{x} \prec \boldsymbol{x}')$ denote the probability that an output of $\boldsymbol{x}$ is also an output of $\boldsymbol{x}'$ i.e., the probability that there exists an edge from $\boldsymbol{x}'$ to the output of $\boldsymbol{x}$ in the bipartite flow network, then we have

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

$$B_N(\mathcal{C}; \mathcal{S}) = \sum_{k=1}^{N} \sum_{\boldsymbol{x}\in\mathcal{C}} \sum_{\boldsymbol{x}'\in\mathcal{C}} Q(\boldsymbol{x} \prec \boldsymbol{x}') = NB_1(\mathcal{C}; \mathcal{S}). \tag{4}$$

Proof: Suppose that the code $\mathcal{C}$ has M codewords $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$ and there are MN random variable outputs, $\mathcal{Y} = \{\boldsymbol{y}_{i,j} : 1 \le i \le M, 1 \le j \le N\}$, where $\boldsymbol{y}_{i,j}$ is the random variable of the j-th output of $\boldsymbol{x}_i$. Note that the expected number of edges in the network $\mathcal{G}_N$ is simply $\sum_{\boldsymbol{x}\in\mathcal{C}} \sum_{\boldsymbol{y}\in\mathcal{Y}} P(\text{there exists an edge from } \boldsymbol{x} \text{ to } \boldsymbol{y})$. And since all the outputs of the channel $\mathcal{S}$ are independent, we have $P(\text{there exists an edge from } \boldsymbol{x}_i \text{ to } \boldsymbol{y}_{j,k}) = Q(\boldsymbol{x}_j \prec \boldsymbol{x}_i)$, for any k. Therefore we have $B_N(\mathcal{C}; \mathcal{S}) = N \sum_{\boldsymbol{x}\in\mathcal{C}} \sum_{\boldsymbol{x}'\in\mathcal{C}} Q(\boldsymbol{x} \prec \boldsymbol{x}') = NB_1(\mathcal{C}; \mathcal{S})$. ■

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 [19].

Proposition 2 [19, Lemma 3]: 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 $\texttt{NP}$-hard [37], we have that evaluating the quantity $B_1(\mathcal{C}; \mathcal{S})$ is also $\texttt{NP}$-hard[4] 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-D, 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).

To this end, we estimate the number of degree-one nodes in $\mathcal{G}_N$.

Proposition 3: The expected number of degree-one right (output) nodes in $\mathcal{G}_N$ is at least $N(M - B_1(\mathcal{C}; \mathcal{S})/2)$.

Proof: We adopt the notation in the proof of Proposition 1. For any output $\boldsymbol{y}_{i,j}$, the expected degree of $\boldsymbol{y}_{i,j}$ (in $\mathcal{G}_N$) is $\sum_{\boldsymbol{x}\in\mathcal{C}} Q(\boldsymbol{x}_i \prec \boldsymbol{x})$). Then by Markov inequality, the probability that $\boldsymbol{y}_{i,j}$ has degree at least two is at most $\frac{1}{2}\sum_{\boldsymbol{x}\in\mathcal{C}} Q(\boldsymbol{x}_i \prec \boldsymbol{x})$. That is, the probability that $\boldsymbol{y}_{i,j}$ has degree one is at least $1 - \frac{1}{2}\sum_{\boldsymbol{x}\in\mathcal{C}} Q(\boldsymbol{x}_i \prec \boldsymbol{x})$. Therefore, the expected number of degree-one right nodes is at least

$$\begin{aligned} &\sum_{i\in[M]} \sum_{j\in[N]} \left(1 - \frac{1}{2}\sum_{\boldsymbol{x}\in\mathcal{C}} Q(\boldsymbol{x}_i \prec \boldsymbol{x})\right) \\ &= N\left(M - \frac{1}{2}\sum_{\boldsymbol{x}'\in\mathcal{C}}\sum_{\boldsymbol{x}\in\mathcal{C}} Q(\boldsymbol{x}' \prec \boldsymbol{x})\right) \\ &= N\left(M - \frac{B_1(\mathcal{C}; \mathcal{S})}{2}\right). \end{aligned}$$ ■

Corollary 1:

$$A^*_N(\mathcal{C}; \mathcal{S}) \le \begin{cases} M + \frac{NB_1(\mathcal{C};\mathcal{S})}{2}, & \text{if } N \ge 2, \\ B_1(\mathcal{C}; \mathcal{S}), & \text{if } N = 1. \end{cases}$$

$$B^*_N(\mathcal{C}; \mathcal{S}) \le N\left(\frac{3}{2}B_1(\mathcal{C}; \mathcal{S}) - M\right).$$

Proof: Recall that all degree-one nodes and their corresponding edges must be removed at the first step. Thus, using Proposition 3, the remaining number of edges is at most $NB_1(\mathcal{C}; \mathcal{S}) - N(M - B_1(\mathcal{C}; \mathcal{S})/2)$ and the remaining number of nodes is at most $M(N+1) - N(M - B_1(\mathcal{C}; \mathcal{S})/2)$. Note in the case for $N = 1$, the expected number of degree-one left nodes is also given by $M - B_1(\mathcal{C}, \mathcal{S})/2$ and these nodes can be removed. ■

For the next two sections, we analyse the enumeration of the expected number of edges for two particular channels, namely the deletion channel and the insertion channel. Firstly,

[4]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.

in Section IV, we focus on the deletion channel, and give a closed expression of the expected number of edges when the codes are all binary words, even-weight codes and constant-weight codes. Furthermore, in Section IV-A, we observe that the edge density of the flow network in the deletion channel is polarized. Since our closed expression only works for certain codes, we give an alternative enumeration method via a dynamic programming approach in Section IV-B that works for any code that can be defined by a *linear sydnrome*. Finally, in Section V, we do a similar analysis for the insertion channel.

IV. Deletion Channel

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

Proposition 4: Fix any code $\mathcal{C} \subseteq \Sigma^n$. We have that

$$B(\mathcal{C}, \text{Del}(p)) = \sum_{t=0}^{n} \sum_{z \in \Sigma^{n-t}} I(z; \mathcal{C}) I^*(z; \mathcal{C}) p^t (1-p)^{n-t}. \quad (5)$$

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

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

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

Since $\sum_{x \in \mathcal{C}} \text{Emb}(x, z)$ and $\sum_{x' \in \mathcal{C}} \mathbb{I}(z \in x')$ yields the quantities $I^*(z; \mathcal{C})$ and $I(z; \mathcal{C})$, respectively, we obtain (14). ■

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

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

The quantity $I(z; \mathcal{C})$ has been studied in other contexts [38], [39]. Of significance, $I(z; \{0, 1\}^n)$ depends on $|z|$ and n only, while $I(z; \mathcal{C}(n, w))$ depends on $|z|$, wt(z), n, and w only. In both cases, this quantity does not depend on the actual string z. Specifically, we have the following proposition.

Proposition 5 [38], [39]: Let $|z| = n - t$. Then,

$$I(z; \mathcal{A}_n) = \sum_{i=0}^{t} \binom{n}{i} \triangleq I_A(n, t). \quad (6)$$

Furthermore, if wt$(z) = u$,

$$\begin{aligned} &I\big(z; \mathcal{C}_{n,w}\big) \\ &\triangleq I_C(n, w, t, u) \\ &= \begin{cases} \binom{n}{w}, & \text{if } u = 0 \text{ and } w \le t, \\ \sum_{i=u}^{t-w+2u} \binom{i-1}{u-1} \binom{n-i}{w-u}, & \text{if } u \ge 1. \end{cases} \end{aligned} \quad (7)$$

Moreover,

$$I(z; \mathcal{E}_n) = \sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} I\big(z; \mathcal{C}_{n,2k}\big). \quad (8)$$

Next, using standard combinatorial identities in enumerative combinatorics [40], we have the following results. For completeness, we also provide the proof below.

Proposition 6: If $|z| = n - t$ and wt$(z) = u$, then

$$I^*(z; \mathcal{A}_n) = 2^t \binom{n}{t}, \quad (9)$$

$$I^*\big(z; \mathcal{C}_{n,w}\big) = \binom{n}{t} \binom{t}{w-u}, \quad (10)$$

$$I^*(z; \mathcal{E}_n) = \begin{cases} 2^{t-1}\binom{n}{t}, & \text{if } t \ge 1, \\ 1, & \text{if } t = 0,\ u \text{ is even}, \\ 0, & \text{if } t = 0,\ u \text{ is odd}. \end{cases} \quad (11)$$

Proof: First, we give a proof of (9). Since $|z| = n - t$, after t insertions, we have a supersequence of length n. Counting multiplicity, the multiset of supersequences of z of length n can be obtained by choosing $(n-t)$ positions out of n to place the bits of z in order and then determining the rest of the t empty slots to be either 0 or 1. Thus the size of the multiset of supersequences is $2^t \binom{n}{n-t} = 2^t \binom{n}{t}$.

Next, we give a proof of (10). Counting multiplicity, the multiset of supersequences of length n of z can be obtained by choosing $(n-t)$ positions to place the bits of z in order, and then choosing $w - u$ positions out of the remaining t empty slots to place 1's. Finally the remaining $t - (w - u)$ positions will be filled with 0's, and thus the supersequence is of length n and weight w. Thus the size of the multiset of supersequence is $\binom{n}{t}\binom{t}{w-u}$.

Finally, we give a proof of (11). The case of $t = 0$ is obvious. Thus we give a proof of when $t \ge 1$. From the previous part, we have $I^*(z; \mathcal{E}_n) = \sum_{w \text{ is even}} I^*(z; \mathcal{C}_{n,w})$. If u is odd, then $I^*(z; \mathcal{E}_n) = \binom{n}{t}\{\binom{t}{1} + \binom{t}{3} + \cdots\} = \binom{n}{t}2^{t-1}$. If u is even, then $I^*(z; \mathcal{E}_n) = \binom{n}{t}\{\binom{t}{0} + \binom{t}{2} + \cdots\} = \binom{n}{t}2^{t-1}$. ■

Using (5)–(10), we then obtain the following closed formulae for the expected number of edges.

Theorem 2: For all n,

$$B(\mathcal{A}_n, \text{Del}(p)) = 2^n \sum_{t=0}^{n} \sum_{i=0}^{t} \binom{n}{i} \binom{n}{t} p^t (1-p)^{n-t}. \quad (12)$$

For $w \le n$,

$$B\big(\mathcal{C}_{n,w}, \mathrm{Del}(p)\big) = \sum_{t=0}^{n}\sum_{u=0}^{w}\binom{n-t}{u}\binom{n}{t}\binom{t}{w-u} I_{\mathcal{C}}(n,w,t,u)p^t(1-p)^{n-t}. \quad (13)$$

Proof: From (5), (6) and (9), we have

$$\begin{aligned} B(\mathcal{A}_n, \mathrm{Del}(p)) &= \sum_{t=0}^{n}\sum_{z\in\Sigma^{n-t}} I(z;\mathcal{A}_n)I^*(z;\mathcal{A}_n)p^t(1-p)^{n-t} \\ &= \sum_{t=0}^{n} 2^t\binom{n}{t}p^t(1-p)^{n-t}\sum_{z\in\Sigma^{n-t}}\sum_{i=0}^{t}\binom{n}{i} \\ &= \sum_{t=0}^{n} 2^t\binom{n}{t}p^t(1-p)^{n-t}2^{n-t}\sum_{i=0}^{t}\binom{n}{i} \\ &= 2^n\sum_{t=0}^{n}\sum_{i=0}^{t}\binom{n}{i}\binom{n}{t}p^t(1-p)^{n-t}. \end{aligned}$$

For $w \le n$, we have from (5), (7) and (10),

$$\begin{aligned} B\big(\mathcal{C}_{n,w}, \mathrm{Del}(p)\big) &= \sum_{t=0}^{n}\sum_{z\in\Sigma^{n-t}} I\big(z;\mathcal{C}_{n,w}\big)I^*\big(z;\mathcal{C}_{n,w}\big)p^t(1-p)^{n-t} \\ &= \sum_{t=0}^{n}\sum_{u=0}^{w}\sum_{z\in\mathcal{C}_{n-t,u}} I\big(z;\mathcal{C}_{n,w}\big)I^*\big(z;\mathcal{C}_{n,w}\big)p^t(1-p)^{n-t} \\ &= \sum_{t=0}^{n}\sum_{u=0}^{w}\binom{n-t}{u}I_{\mathcal{C}}(n,w,t,u)\binom{n}{t}\binom{t}{w-u}p^t(1-p)^{n-t}. \end{aligned}$$

■

Now, for the codebook with even-weight words, one can apply (8) and (11) 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.

Lemma 1: If n is odd,

$$B(\mathcal{E}_n, \mathrm{Del}(p)) = \frac{1}{4}B(\mathcal{A}_n, \mathrm{Del}(p)) + 2^{n-2}(1-p)^n. \quad (14)$$

To prove this lemma, we use the following combinatorial proposition.

Proposition 7: Let $0 \le t \le n$. If n is odd, then

$$\sum_{z\in\{0,1\}^{n-t}} I(z;\mathcal{E}_n) = \frac{1}{2}\sum_{z\in\{0,1\}^{n-t}} I(z;\mathcal{A}_n). \quad (15)$$

Proof: For $z \in \{0,1\}^{n-t}$, we let $\bar{z}$ be the complement of z. We consider the codebook $\mathcal{O}_n \triangleq \{x \in \{0,1\}^n : \mathrm{wt}(x) \text{ is odd}\}$, and the sets $\mathcal{I}(z;\mathcal{E}_n) = \{x \in \mathcal{E}_n : z \in x\}$ and $\mathcal{I}(\bar{z};\mathcal{O}_n) = \{x \in \mathcal{O}_n : \bar{z} \in x\}$. Recall that $z \in x$ if and only if z is a subsequence of x. Since $x \in \mathcal{I}(z;\mathcal{E}_n)$ if and only if $\bar{x} \in \mathcal{I}(\bar{z};\mathcal{O}_n)$. The two sets have the same cardinality.

Therefore,

$$I(z;\mathcal{E}_n) + I(\bar{z};\mathcal{E}_n) = I(z;\mathcal{E}_n) + I(z;\mathcal{O}_n) = I(z,\mathcal{A}_n).$$

Summing this equation over all $z \in \{0,1\}^{n-t}$, we obtain (15). ■

Proof of Lemma 1: It is immediate from (9), (11) and (15) that for $t \ge 1$,

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

So, we have that

$$\begin{aligned} B_1(\mathcal{E}_n, \mathrm{Del}(p)) &= \frac{1}{4}\big(B(\mathcal{A}_n, \mathrm{Del}(p)) - 2^n(1-p)^n\big) + 2^{n-1}(1-p)^n \\ &= \frac{1}{4}B_1(\mathcal{A}_n, \mathrm{Del}(p)) + 2^{n-2}(1-p)^n. \end{aligned}$$ ■

Unfortunately, we are unable to obtain similar expression to the case where n is even. Nevertheless, as we observe in Figure 1(b), the edge density polarizes in a fashion similar to the case where n is odd.

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 $\lim_{n\to\infty} B(\mathcal{C}_n, \mathrm{Del}(p))/|\mathcal{C}_n|^2$. Observe that $B(\mathcal{C}_n, \mathrm{Del}(p))/|\mathcal{C}_n|^2$ 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 8: Let $\mathcal{C} = \mathcal{A}_n$ and $M = 2^n$. For $0 \le p < 1/2$, set $\beta = \sqrt{p}/(\sqrt{p} + \sqrt{1-p})$ and choose $\epsilon > H(\beta)$. Then for sufficiently large n, we have that $B(\mathcal{A}_n, \mathrm{Del}(p)) \le M^{1+\epsilon} = o(M^2)$.

Proof: Changing the order of summation in (12), we have that

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

Set $t' = \lfloor (n+1)p \rfloor$ and we upper bound the above sum in two parts. Specifically, for $0 \le i \le t'$, we show that the sum is at most $2^{(1+H(p))n}$, while for $t'+1 \le i \le n$, we show that the sum is at most $n^2 2^{(H(\beta)+1)n}$ with β as defined in the proposition.

First, for $0 \le i \le t'$, we have

$$\binom{n}{i}\sum_{t=i}^{n}\binom{n}{t}p^t(1-p)^{n-t} \le \binom{n}{i}\sum_{t=0}^{n}\binom{n}{t}p^t(1-p)^{n-t} = \binom{n}{i}.$$

Hence,

$$\begin{aligned} 2^n\sum_{i=0}^{t'}\binom{n}{i}\sum_{t=i}^{n}\binom{n}{t}p^t(1-p)^{n-t} &\le 2^n\sum_{i=0}^{t'}\binom{n}{i} \\ &\le 2^{(1+H(p))n}, \end{aligned} \quad (16)$$

as required. Next, it follows from standard facts of the binomial distribution that for all t,

$$\binom{n}{t}p^t(1-p)^{n-t} \le \binom{n}{t'}p^{t'}(1-p)^{n-t'} \le 1, \quad (17)$$

Furthermore, we have that

$$\binom{n}{t}p^t(1-p)^{n-t} \ge \binom{n}{s}p^s(1-p)^{n-s} \text{ for all } s \ge t \ge t'. \quad (18)$$

So, for $t'+1 \le i \le n$,

$$\binom{n}{i}\sum_{t=i}^{n}\binom{n}{t}p^t(1-p)^{n-t} \le n\binom{n}{i}\binom{n}{i}p^i(1-p)^{n-i},$$

we have that

$$2^n\sum_{i=t'+1}^{n}\binom{n}{i}\sum_{t=i}^{n}\binom{n}{t}p^t(1-p)^{n-t} \le n2^n\sum_{i=0}^{n}\binom{n}{i}^2 p^i(1-p)^{n-i}.$$

Now, since $\log_2(\binom{n}{i}^2 p^i(1-p)^{n-i})$ is at most $(2H(i/n) + (i/n)\log_2 p + (1-i/n)\log_2(1-p))n$, we maximize the function $2H(x)+x\log_2 p+(1-x)\log_2(1-p)$ in the interval $0 \le x \le 1$. By considering its first derivative, we have that the function is maximized when $x = \sqrt{p}/(\sqrt{p}+\sqrt{1-p}) \triangleq \beta$. Then for this value of β, we have that $\binom{n}{\beta n}^2 p^{\beta n}(1-p)^{n-\beta n} \le \binom{n}{\beta n} \le 2^{H(\beta)n}$. Therefore,

$$n2^n\sum_{i=0}^{n}\binom{n}{i}^2 p^i(1-p)^{n-i} \le n^2 2^n \cdot 2^{H(\beta)n} = n^2 2^{(H(\beta)+1)n}. \quad (19)$$

Finally, since $p < 1/2$, via standard manipulations, we have that $p < \beta < 1/2$ and so, $H(p) < H(\beta) < 1$. Then, combining both (16) and (19), we have that the expected number of edges is at most $2^{(1+\epsilon)n} = M^{(1+\epsilon)}$ for any $\epsilon > H(\beta)$ and sufficiently large n. ■

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 $B_1(\mathcal{A}_n, \mathrm{Del}(p)) = o(M^2)$ and when $p > 1/2$, we have $B_1(\mathcal{A}_n, \mathrm{Del}(p))$ approaches M^2. Before we formally state the result, we also observe that $B_1(\mathcal{E}_n, \mathrm{Del}(p))$ share the same polarization behavior as $B_1(\mathcal{A}_n, \mathrm{Del}(p))$. This follows directly from (14).

Proposition 9:

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

Proof:

- When $p < 1/2$, the limit value is immediate from Proposition 8.
- When $p = 1/2$, the expression (12) reduces to

$$B(\mathcal{A}_n, \mathrm{Del}(p)) = \sum_{t=0}^{n}\sum_{i=0}^{t}\binom{n}{t}\binom{n}{i} \quad (20)$$

Note that by changing the order of summation, we have that

$$\begin{aligned}B(\mathcal{A}_n, \mathrm{Del}(p)) &= \sum_{i=0}^{n}\sum_{t=i}^{n}\binom{n}{t}\binom{n}{i}\\ &= \sum_{t=0}^{n}\sum_{i=t}^{n}\binom{n}{t}\binom{n}{i}. \quad (21)\end{aligned}$$

The second equality results from the renaming of variables. Then adding (20) and (21), we have

$$2B(\mathcal{A}_n, \mathrm{Del}(p)) = \sum_{i=0}^{n}\sum_{t=0}^{n}\binom{n}{t}\binom{n}{i} + \sum_{t=0}^{n}\binom{n}{t}^2.$$

Now, $\sum_{i=0}^{n}\sum_{t=0}^{n}\binom{n}{t}\binom{n}{i} = (\sum_{t=0}^{n}\binom{n}{t})^2 = 2^{2n}$, while $\sum_{t=0}^{n}\binom{n}{t}^2 = \binom{2n}{n}$. Since $\binom{2n}{n} \sim 2^{2n}/\sqrt{\pi n}$, we have that $\lim_{n\to\infty}\binom{2n}{n}/2^{2n} = \lim_{n\to\infty}1/\sqrt{\pi n} = 0$. Therefore, $\lim_{n\to\infty}2B(\mathcal{A}_n, \mathrm{Del}(p))/2^{2n} = 1$ and so, $\lim_{n\to\infty}B(\mathcal{A}_n, \mathrm{Del}(p))/2^{2n} = 1/2$.
- When $p > 1/2$, we rewrite the expression (12) as

$$\begin{aligned}&B(\mathcal{A}_n, \mathrm{Del}(p))\\ &= 2^n\sum_{i=0}^{n}\binom{n}{i}\sum_{t=0}^{n}\binom{n}{t}p^t(1-p)^{n-t}\\ &\quad -2^n\sum_{i=0}^{n}\binom{n}{i}\sum_{t=0}^{i-1}\binom{n}{t}p^t(1-p)^{n-t}. \quad (22)\end{aligned}$$

For the first summand, we have

$$2^n\sum_{i=0}^{n}\binom{n}{i}\sum_{t=0}^{n}\binom{n}{t}p^t(1-p)^{n-t} = 2^n\sum_{i=0}^{n}\binom{n}{i} = 2^{2n}. \quad (23)$$

For the second summand, we change variables by setting $t' = n - t$ and $i' = n - i$ and we have

$$\begin{aligned}&2^n\sum_{i=0}^{n}\binom{n}{i}\sum_{t=0}^{i-1}\binom{n}{t}p^t(1-p)^{n-t}\\ &= 2^n\sum_{i=0}^{n}\binom{n}{i}\sum_{t'=n-i+1}^{n}\binom{n}{t'}p^{n-t'}(1-p)^{t'}\\ &= 2^n\sum_{i'=0}^{n}\binom{n}{i'}\sum_{t'=i'+1}^{n}\binom{n}{t'}p^{n-t'}(1-p)^{t'}\\ &\le 2^n\sum_{i'=0}^{n}\binom{n}{i'}\sum_{t'=i'}^{n}\binom{n}{t'}p^{n-t'}(1-p)^{t'}. \quad (24)\end{aligned}$$

Since $p > 1/2$, we have that $(1-p) < 1/2$ and then Proposition 8 implies that the second summand (24) is $o(M^2)$. The proposition is then immediate from (22) and (23). ■

Here, we conjecture that the same polarization phenomenon is present for constant-weight codebooks.

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

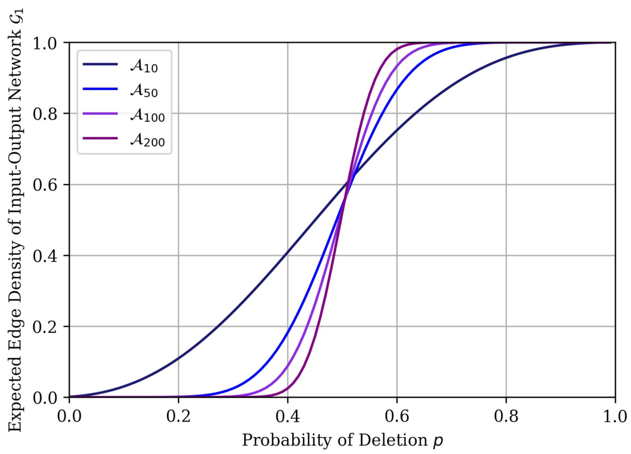
$$\lim_{n\to\infty}\frac{B_1(\mathcal{C}_{n,\lfloor\omega n\rfloor}, \mathrm{Del}(p))}{|\mathcal{C}_{n,\lfloor\omega n\rfloor}|^2} = \begin{cases} 0, & \text{if } p < p_\omega,\\ 1, & \text{if } p > p_\omega.\end{cases}$$

In Figure 1, we exhibit this polarization phenomenon numerically. Specifically, Figure 1(a) and (c) corroborate with Proposition 9 and Conjecture 1, respectively.

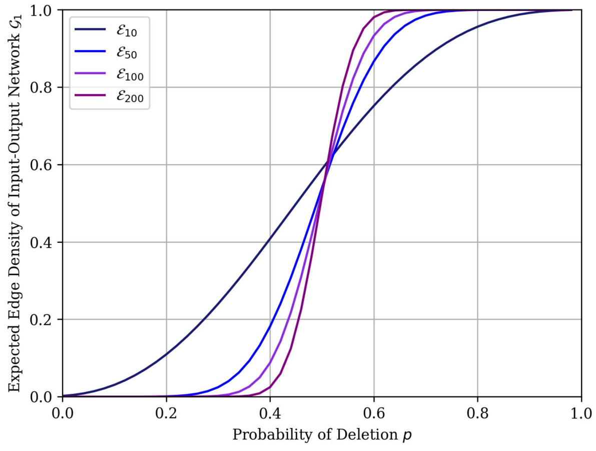
B. Enumeration via Dynamic Programming

Here, we consider the problem of enumerating $\mathcal{B}_1(\mathcal{C}, \mathrm{Del}(p))$ for a certain class of codebooks. Consider a finite ring R that contains the q-ary alphabet Σ. We fix $\boldsymbol{H}$ to

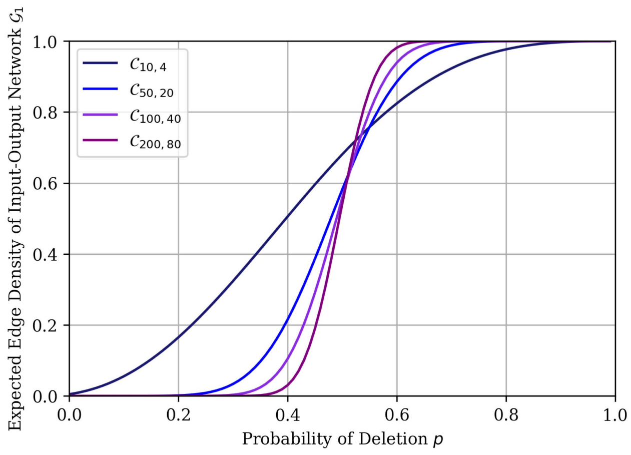
(a) Codebooks with all binary words $\mathcal{A}_n$ over deletion channels



(b) Even-weight codebooks $\mathcal{E}_n$ over deletion channels



(c) Constant weight codebooks $\mathcal{C}_{n,\omega n}$ with $\omega = 0.4$ over deletion channels



(d) Codebooks with all binary words $\mathcal{A}_n$ over insertion channels

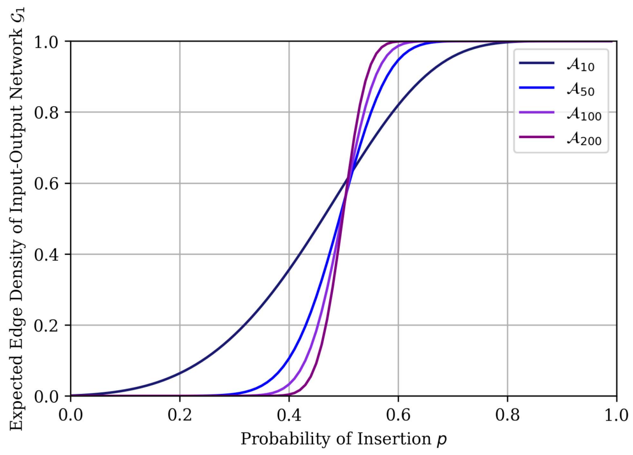


Fig. 1. Polarization behavior of expected edge density for certain codebooks over deletion and insertion channels. Exact formulas for edge densities are given in Theorems 2 and 3. Figure 1(d) is obtained from Theorem 3 in Section V.

be a $r \times n$-matrix over R and $\boldsymbol{\sigma}$ to be some syndrome in R^r. Then we consider the codebook $\mathcal{C}$ is defined to where

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

If a codebook satisfies this definition, we say that the codebook is *defined by a linear syndrome* and we remark 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 resulting codebook $\mathcal{C}$ is the Varshamov-Tenengolts code [35] that corrects a single deletion.
- 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.

Before we describe the detailed recursive formulae and the dynamic programming implementation, we first state the running time of our proposed enumeration method.

Proposition 10: Let $\mathcal{C}$ be the q-ary code defined by a linear syndrome with ring R, $r \times n$-matrix $\boldsymbol{H}$ and syndrome $\boldsymbol{\sigma}$. Then $B(\mathcal{C}, \mathrm{Del}(p))$ can be determined in $O(n^{2q}|R|^{2r})$ time. In particular, if $\mathcal{C}$ is a VT code of length n, we can determine $B(\mathcal{C}, \mathrm{Del}(p))$ in $O(n^6)$ time.

The proof of Proposition 10 will be discussed later in the paper, but for now, we describe our enumeration method. Let the columns of $\boldsymbol{H}$ be $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n$. In other words, $\boldsymbol{H} = (\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n)$. For $\boldsymbol{z} \in \Sigma^\ell$ with $\ell \le n$, we define its ℓ-th partial syndrome $S_\ell(\boldsymbol{z}) = \boldsymbol{z}\boldsymbol{H}_\ell^T$, where $\boldsymbol{H}_\ell = (\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_\ell)$. Note that $S_\ell(\boldsymbol{z}) \in R^r$.

To simplify our exposition, we describe our method for the binary alphabet $\Sigma = \{0, 1\}$ only. The method can be easily extended for nonbinary alphabet. First, we partition the set of binary strings according to the positions of their last one and/or zero. Let $m \le n$. Define $\mathcal{B}(m, \ell_0, \ell_1, \boldsymbol{\alpha})$ to be the set of binary strings $\boldsymbol{x}$ of length m whose last index is zero and one are ℓ_0 and ℓ_1, respectively, such that $S_m(\boldsymbol{x}) = \boldsymbol{\alpha}$. Observe that we have either $\ell_0 = m$ and $0 \le \ell_1 \le m - 1$ or vice versa. Here, we adopt the convention that $\ell_0 = 0$ or $\ell_1 = 0$ if the string contains all ones or all zeroes, respectively. For simplicity, we define $G(m, \boldsymbol{\alpha})$ to be the number of binary strings $\boldsymbol{x}$ of length m such that $S_m(\boldsymbol{x}) = \boldsymbol{\alpha}$. Therefore, we have the following recursion $G(m, \boldsymbol{\alpha}) = G(m-1, \boldsymbol{\alpha}) + G(m-1, \boldsymbol{\alpha} - \boldsymbol{h}_m)$, where $G(0, \boldsymbol{\alpha}) = 1$, if $\boldsymbol{\alpha} = \boldsymbol{0}$; and $G(0, \boldsymbol{\alpha}) = 0$, otherwise.

Next, we consider the quantity

$$T(i,a,\boldsymbol{\alpha};j,\ell_0,\ell_1,\boldsymbol{\beta}) \triangleq \sum_{\substack{|\boldsymbol{x}|=i,\\ x_i=a,\\ S_i(x)=\boldsymbol{\alpha}}} \sum_{\boldsymbol{y}\in\mathcal{B}(j,\ell_0,\ell_1,\boldsymbol{\beta})} Q(\boldsymbol{x}\prec\boldsymbol{y}). \quad (25)$$

To keep our notations succinct, we adopt the following abbreviations.

$$T(i,a,\boldsymbol{\alpha};j,*,\ell_1,\boldsymbol{\beta}) = \begin{cases}\sum_{k=0}^{j-1} T(i,a,\boldsymbol{\alpha};j,k,j,\boldsymbol{\beta}), & \text{if } \ell_1=j>0,\\ T(i,a,\boldsymbol{\alpha};j,j,\ell_1,\boldsymbol{\beta}), & \text{if } \ell_1<j,\end{cases}$$

$$T(i,a,\boldsymbol{\alpha};j,\ell_0,*,\boldsymbol{\beta}) = \begin{cases}\sum_{k=0}^{j-1} T(i,a,\boldsymbol{\alpha};j,j,k,\boldsymbol{\beta}), & \text{if } \ell_0=j>0,\\ T(i,a,\boldsymbol{\alpha};j,\ell_0,j,\boldsymbol{\beta}), & \text{if } \ell_0<j,\end{cases}$$

$$T(i,a,\boldsymbol{\alpha};j,*,*,\boldsymbol{\beta}) = T(i,a,\boldsymbol{\alpha};j,*,j,\boldsymbol{\beta}) + T(i,a,\boldsymbol{\alpha};j,j,*,\boldsymbol{\beta}), \text{ if } j>0,$$

$$T(i,*,\boldsymbol{\alpha};j,\ell_0,\ell_1,\boldsymbol{\beta}) = T(i,0,\boldsymbol{\alpha};j,\ell_0,\ell_1,\boldsymbol{\beta}) + T(i,1;j,\ell_0,\ell_1,\boldsymbol{\beta}), \text{ if } i>0,$$

$$T(i,*,\boldsymbol{\alpha};j,\ell_0,*,\boldsymbol{\beta}) = T(i,0,\boldsymbol{\alpha};j,\ell_0,*,\boldsymbol{\beta}) + T(i,1,\boldsymbol{\alpha};j,\ell_0,*,\boldsymbol{\beta}), \text{ if } i>0,$$

$$T(i,*,\boldsymbol{\alpha};j,*,\ell_1,\boldsymbol{\beta}) = T(i,0,\boldsymbol{\alpha};j,*,\ell_1,\boldsymbol{\beta}) + T(i,1,\boldsymbol{\alpha};j,*,\ell_1,\boldsymbol{\beta}), \text{ if } i>0,$$

$$T(i,*,\boldsymbol{\alpha};j,*,*,\boldsymbol{\beta}) = T(i,0,\boldsymbol{\alpha};j,*,*,\boldsymbol{\beta}) + T(i,1,\boldsymbol{\alpha};j,*,*,\boldsymbol{\beta}), \text{ if } i>0,$$

$$\mathcal{B}(j,\ell_0,*,\boldsymbol{\beta}) = \begin{cases}\bigcup_{k=0}^{j-1}\mathcal{B}(j,j,k,\boldsymbol{\beta}), & \text{if } \ell_0=j>0,\\ \mathcal{B}(j,\ell_0,j,\boldsymbol{\beta}), & \text{if } \ell_0<j,\end{cases}$$

$$\mathcal{B}(j,*,\ell_1,\boldsymbol{\beta}) = \begin{cases}\bigcup_{k=0}^{j-1}\mathcal{B}(j,k,j,\boldsymbol{\beta}), & \text{if } \ell_1=j>0,\\ \mathcal{B}(j,j,\ell_1,\boldsymbol{\beta}), & \text{if } \ell_1<j,\end{cases}$$

$$\mathcal{B}(j,*,*,\boldsymbol{\beta}) = \mathcal{B}(j,j,*,\boldsymbol{\beta}) \cup \mathcal{B}(j,*,j,\boldsymbol{\beta}).$$

Recall that $\mathcal{C}$ comprises all words $\boldsymbol{x}$ of length n with $S_n(\boldsymbol{x}) = \boldsymbol{\sigma}$. Hence, the quantity of interest $B(\mathcal{C},\text{Del}(p))$ is given by $T(n,*,\boldsymbol{\sigma};n,*,*,\boldsymbol{\sigma})$. The following base cases are trivial and hence we state them without proof.

Lemma 2 (Base Cases): In what follows, we use the symbol ? to represent any symbol, i.e., (?, ?) can be either $(*,*)$ or $(\ell_0,*)$ or $(*,\ell_1)$, or (ℓ_0,ℓ_1).

1) When $i=0$, we have the following.
   a) If $j<0$, then $T(0,a,\boldsymbol{\alpha};j,\ell_0,\ell_1,\boldsymbol{\beta})=0$.
   b) If $j\geq 0$, then $T(0,*,\boldsymbol{0};j,?,?,\boldsymbol{\beta}) = |\mathcal{B}(j,?,?,\boldsymbol{\beta})|$ and $T(0,*,\boldsymbol{\alpha};j,?,?,\boldsymbol{\beta})=0$, if $\boldsymbol{\alpha}\neq\boldsymbol{0}$.
2) When $i>0$, we have the following.
   a) $T(i,*,\boldsymbol{\alpha};0,?,?,\boldsymbol{\beta})=0$, if $\boldsymbol{\beta}\neq\boldsymbol{0}$.
   b) $T(i,*,\boldsymbol{\alpha};0,?,?,\boldsymbol{0}) = p^i|\{\boldsymbol{x}\in\{0,1\}^i : S_i(\boldsymbol{x})=\boldsymbol{\alpha}\}| = p^i|\mathcal{B}(i,*,*,\boldsymbol{\alpha})| = p^iG(i,\boldsymbol{\alpha})$.

Now, to compute the size of $\mathcal{B}$, we use the following lemma.

Lemma 3 (Size of $\mathcal{B}$): As before, we use the symbol ? to represent any symbol, i.e., (?, ?) can be either $(*,*)$ or $(\ell_0,*)$ or $(*,\ell_1)$, or (ℓ_0,ℓ_1).

1) If $j=0$, then $|\mathcal{B}(0,?,?,\boldsymbol{0})|=1$ and $|\mathcal{B}(0,?,?,\boldsymbol{\beta})|=0$, if $\boldsymbol{\beta}\neq\boldsymbol{0}$.
2) If $j>0$, then

$$|\mathcal{B}(j,*,*,\boldsymbol{\beta})| = G(j,\boldsymbol{\beta}),$$

$$|\mathcal{B}(j,\ell_0,*,\boldsymbol{\beta})| = \begin{cases} G(\ell_0-1,\boldsymbol{\beta}-\boldsymbol{h}_j-\boldsymbol{h}_{j-1}-\cdots-\boldsymbol{h}_{\ell_0+1}), & \text{if } 0<\ell_0\leq j,\\ G(0,\boldsymbol{\beta}-\boldsymbol{h}_j-\boldsymbol{h}_{j-1}-\cdots-\boldsymbol{h}_1), & \text{if } 0=\ell_0<j.\end{cases}$$

$$|\mathcal{B}(j,*,\ell_1,\boldsymbol{\beta})| = \begin{cases} G(\ell_1-1,\boldsymbol{\beta}-\boldsymbol{h}_{\ell_1}), & \text{if } 0<\ell_1\leq j,\\ G(0,\boldsymbol{\beta}), & \text{if } 0=\ell_1<j.\end{cases}$$

$$|\mathcal{B}(j,\ell_0,\ell_1,\boldsymbol{\beta})| = \begin{cases} |\mathcal{B}(j,*,\ell_1,\boldsymbol{\beta})|=G(\ell_1-1,\boldsymbol{\beta}-\boldsymbol{h}_{\ell_1}) & \text{if } \ell_0>\ell_1>0,\\ G(0,\boldsymbol{\beta}) & \text{if } \ell_0>\ell_1=0.\end{cases}$$

$$|\mathcal{B}(j,\ell_0,\ell_1,\boldsymbol{\beta})| = \begin{cases} |\mathcal{B}(j,\ell_0,*,\boldsymbol{\beta})|=G(\ell_0-1,\boldsymbol{\beta}-\boldsymbol{h}_j-\boldsymbol{h}_{j-1}-\cdots-\boldsymbol{h}_{\ell_0+1}),\\ \text{if } 0<\ell_0<\ell_1,\\ |\mathcal{B}(j,\ell_0,*,\boldsymbol{\beta})|=G(0,\boldsymbol{\beta}-\boldsymbol{h}_1-\boldsymbol{h}_2-\cdots-\boldsymbol{h}_j)\\ \text{if } 0=\ell_0<\ell_1.\end{cases}$$

Next, we demonstrate the following recursion rules.

Lemma 4 (Recursion Rules): Let $i>0$ and $j>0$. We have the following recursion rules.

1) If $a=0$, $\ell_0=j$, then $T(i,0,\boldsymbol{\alpha};j,j,\ell_1,\boldsymbol{\beta}) = pT(i-1,*,\boldsymbol{\alpha};j,j,\ell_1,\boldsymbol{\beta}) + (1-p)T(i-1,*,\boldsymbol{\alpha};j-1,*,\ell_1,\boldsymbol{\beta})$.
2) If $a=0$, $\ell_1=j$, then $T(i,0,\boldsymbol{\alpha};j,\ell_0,j,\boldsymbol{\beta}) = pT(i-1,*,\boldsymbol{\alpha};j,\ell_0,j,\boldsymbol{\beta}) + (1-p)T(i-1,*,\boldsymbol{\alpha};\ell_0-1,*,*,\boldsymbol{\beta}-(\sum_{k=\ell_0+1}^{j}\boldsymbol{h}_k))$.
3) If $a=1$, $\ell_0=j$, then $T(i,1,\boldsymbol{\alpha};j,j,\ell_1,\boldsymbol{\beta}) = pT(i-1,*,\boldsymbol{\alpha}-\boldsymbol{h}_i;j,j,\ell_1,\boldsymbol{\beta}) + (1-p)T(i-1,*,\boldsymbol{\alpha}-\boldsymbol{h}_i;\ell_1-1,*,*,\boldsymbol{\beta}-\boldsymbol{h}_{\ell_1})$.
4) If $a=1$, $\ell_1=j$, then $T(i,1,\boldsymbol{\alpha};j,\ell_0,j,\boldsymbol{\beta}) = pT(i-1,*,\boldsymbol{\alpha}-\boldsymbol{h}_i;j,\ell_0,j,\boldsymbol{\beta}) + (1-p)T(i-1,*,\boldsymbol{\alpha}-\boldsymbol{h}_i;j-1,\ell_0,*,\boldsymbol{\beta}-\boldsymbol{h}_j)$.

Proof: For 1), observe that $T(i,0,\boldsymbol{\alpha};j,j,\ell_1,\boldsymbol{\beta})$ is the sum of $Q(\boldsymbol{x}\prec\boldsymbol{y})$ for all $\boldsymbol{x}\in\mathcal{B}(i,i,*,\boldsymbol{\alpha})$ and $\boldsymbol{y}\in\mathcal{B}(j,j,\ell_1,\boldsymbol{\beta})$. Recall that each bit in $\boldsymbol{x}$ is independently deleted with probability p. So, we analyze according to the last bit of $\boldsymbol{x}$, and perform recursion on the remaining $i-1$ bits. Given that the last bit of $\boldsymbol{x}$ is deleted, then $\sum_{\boldsymbol{x}\in\mathcal{B}(i,i,*,\boldsymbol{\alpha})}\sum_{\boldsymbol{y}\in\mathcal{B}(j,j,\ell_1,\boldsymbol{\beta})} Q(\boldsymbol{x}\prec\boldsymbol{y}) = \sum_{\boldsymbol{x}\in\mathcal{B}(i-1,*,*,\boldsymbol{\alpha})}\sum_{\boldsymbol{y}\in\mathcal{B}(j,j,\ell_1,\boldsymbol{\beta})} Q(\boldsymbol{x}\prec\boldsymbol{y}) = T(i-1,*,\boldsymbol{\alpha};j,\ell_0,\ell_1,\boldsymbol{\beta})$.

On the other hand, if the last bit of $\boldsymbol{x}=x_1x_2\dots x_i$ is not deleted, then the output of $\boldsymbol{x}$ is a subsequence of $\boldsymbol{y}$ if and only if the output of $x_1x_2\dots x_{i-1}$ is a subsequence of $y_1y_2\dots y_{j-1}$, since $\boldsymbol{y}$ ends with $0=x_i$. Thus given that x_i is not deleted, we have $Q(\boldsymbol{x}\prec\boldsymbol{y}) = Q(x_1x_2\dots x_{i-1}\prec y_1y_2,\dots y_{j-1})$. Thus, given that the last bit of $\boldsymbol{x}$ is not deleted, we have that $\sum_{\boldsymbol{x}\in\mathcal{B}(i,i,*,\boldsymbol{\alpha})}\sum_{\boldsymbol{y}\in\mathcal{B}(j,j,\ell_1,\boldsymbol{\beta})} Q(\boldsymbol{x}\prec\boldsymbol{y}) = \sum_{\boldsymbol{x}\in\mathcal{B}(i-1,*,*,\boldsymbol{\alpha})}\sum_{\boldsymbol{y}\in\mathcal{B}(j-1,*,\ell_1,\boldsymbol{\beta})} Q(\boldsymbol{x}\prec\boldsymbol{y}) = T(i-1,*,\boldsymbol{\alpha};j-1,*,\ell_1,\boldsymbol{\beta})$. Finally, combining the two cases, we obtain recursion rule 1).

For 2), observe that $T(i,0,\boldsymbol{\alpha};j,\ell_0,j,\boldsymbol{\beta})$ is the sum of $Q(\boldsymbol{x}\prec\boldsymbol{y})$ for all $\boldsymbol{x}\in\mathcal{B}(i,i,*,\boldsymbol{\alpha})$ and $\boldsymbol{y}\in\mathcal{B}(j,\ell_0,j,\boldsymbol{\beta})$. Similarly, we can split away the analysis of the output of the last bit of $\boldsymbol{x}$, and do recursion on the remaining $i-1$ bits. Given that the last bit of $\boldsymbol{x}$ is deleted, then $\sum_{\boldsymbol{x}\in\mathcal{B}(i,i,*,\boldsymbol{\alpha})}\sum_{\boldsymbol{y}\in\mathcal{B}(j,j,\ell_1,\boldsymbol{\beta})} Q(\boldsymbol{x}\prec\boldsymbol{y}) = T(i-1,*,\boldsymbol{\alpha};j,\ell_0,\ell_1,\boldsymbol{\beta})$. If the last bit of $\boldsymbol{x}=x_1x_2,\dots x_i$ is not deleted, then the output of $\boldsymbol{x}$ is a subsequence of $\boldsymbol{y}$ if and only if the output of $x_1x_2\dots x_{i-1}$ is a subsequence of $y_1y_2\dots y_{\ell_0-1}$, since the last occurrence of 0 in $\boldsymbol{y}$ is at y_{ℓ_0}. Therefore, given that x_i is not deleted, $Q(\boldsymbol{x}\prec\boldsymbol{y}) = Q(x_1x_2\dots x_{i-1}\prec y_1y_2\dots y_{\ell_0-1})$. Thus, given that the last bit of $\boldsymbol{x}$ is not deleted, we have $\sum_{\boldsymbol{x}\in\mathcal{B}(i,i,*,\boldsymbol{\alpha})}\sum_{\boldsymbol{y}\in\mathcal{B}(j,\ell_0,j,\boldsymbol{\beta})} Q(\boldsymbol{x}\prec\boldsymbol{y}) = \sum_{\boldsymbol{x}\in\mathcal{B}(i-1,*,*,\boldsymbol{\alpha})}\sum_{\boldsymbol{y}\in\mathcal{B}(\ell_0-1,*,*,\boldsymbol{\beta}-\sum_{k=\ell_0+1}^{j}\boldsymbol{h}_k)} Q(\boldsymbol{x}\prec\boldsymbol{y}) = T(i-1,*,\boldsymbol{\alpha};\ell_0-1,*,*,\boldsymbol{\beta}-\sum_{k=\ell_0+1}^{j}\boldsymbol{h}_k)$. Finally, combining the two cases, we recursion rule 2).

The proof of 3) is similar to 2), and the proof of 4) is similar to 1). ■

Finally, we provide a running time analysis for our enumeration method and complete the proof of Proposition 10.

Proof of Proposition 10: For $\Sigma_q=\{0,1,\dots,q-1\}$, we are interested in $T(n,*,\boldsymbol{\sigma};n,*,*,\dots,*,\boldsymbol{\sigma})$ that represents

$$\sum_{a\in\Sigma_q}\sum_{\substack{0\leq\ell_0,\ell_1,\dots,\ell_{q-1}\leq n,\\ \text{where } \ell_k=n \text{ for some } 0\leq k\leq q-1}} T(n,a,\boldsymbol{\sigma};n,\ell_0,\ell_1,\dots,\ell_{q-1},\boldsymbol{\sigma}).$$

As with typical analysis for dynamic programming algorithms, we first determine the total number of sub-problems, that is, the total number of possible inputs for T.

- There are q possibilities for the second argument of T.
- The first and second powers of n come from the possibilities for first and fourth arguments of T.
- The power of $q-1$ of n comes from the possibilities for ℓ_0 until ℓ_{q-1}, except that one of them is fixed to be n.
- Finally, the third and last arguments of T each has $|R|^r$ possibilities.

Hence, in total, we compute $O(n^{q+1}|R|^{2r})$ possible inputs for T.

Next, we need to determine the running time for each input. In all cases in Lemma 4, a recursion rule involves $O(n^{q-1})$ summands. For example, suppose that $a = 0$ and ℓ_0 is the smallest among all ℓ_k, then the q-ary generalization for the recursion rule in Lemma 4, $T(i, 0, \boldsymbol{\alpha}; j, \ell_0, \ell_1, \ldots, \ell_{q-1}, \boldsymbol{\beta})$ has q summands for the first part $pT(i-1, *, \boldsymbol{\alpha}; j, \ell_0, \ell_1, \ldots, \ell_{q-1}, \boldsymbol{\beta})$ and $O(n^{q-1})$ summands for the second part $(1-p)T(i-1, *, \boldsymbol{\alpha}; \ell_0 - 1, *, *, \ldots, *, \boldsymbol{\beta}')$ for some $\boldsymbol{\beta}'$. Thus in total, the time complexity of the algorithm is $O(n^{2q}|R|^{2r})$. ■

V. Insertion Channels

Throughout this section, we have that $\mathcal{S} = \mathrm{Ins}(p)$. Similar to Section IV, the quantity $B_1(\mathcal{C}; \mathcal{S})$ is useful for providing estimates on sizes of the networks $\mathcal{G}_N$ and $\mathcal{G}_N^*$. Hence, we study $B_1(\mathcal{C}; \mathrm{Ins}(p))$, and for brevity, we write this quantity as $B(\mathcal{C}; \mathrm{Ins}(p))$. Our first proposition states that the problem of determining $B(\mathcal{C}, \mathrm{Ins}(p))$ is equivalent to certain enumeration problems concerning supersequences.

Proposition 11: Fix some code $\mathcal{C} \subseteq \Sigma^n$. We have that

$$B(\mathcal{C}, \mathrm{Ins}(p)) = \sum_{t=0}^{\infty} \sum_{z \in \Sigma^{n+t}} D(z; \mathcal{C}) D^*(z; \mathcal{C}) (p/2)^t (1-p)^{n+1}. \quad (26)$$

Here, $D(z; \mathcal{C})$ denotes the number of words in $\mathcal{C}$ that are subsequences[5] of z, while $D^*(z; \mathcal{C}) = \sum_{x \in \mathcal{C}} \mathrm{Emb}(z, x)$. In other words, $D^*(z; \mathcal{C})$ counts the total number of occurrences of all codewords in $\mathcal{C}$ as a subsequence of z.

Proof: Observe for the $\mathrm{Ins}(p)$-channel, the quantity $Q(x \prec x')$ denotes the probability that an output of x is a supersequence of x'. So, for $t \geq 0$, if we use $I_t(x)$ to denote the set of all $(n+t)$-supersequences of x, then we have $Q(x \prec x') = \sum_{t=0}^{\infty} \sum_{z \in I_t(x)} \mathrm{Emb}(z, x) \mathbb{I}(x' \in z)(p/2)^t (1-p)^{n+1}$. This is because for each $z \in I_t(x)$, the probability of x being a specific embedding of z is to have $n+1$ **Correct** events based on Definition 1, and t **Insertion** events, each with probability $\frac{1}{2}p$ (because each bit may be inserted with equal probability). Here, we use $\mathbb{I}(x' \in z)$ to denote the indicator function for the event that x' is a subsequence of z. Using this expression and switching the order of summation, we can rewrite (3) as

$$\begin{aligned} &B(\mathcal{C}, \mathrm{Ins}(p)) \\ &= \sum_{t=0}^{\infty} (p/2)^t (1-p)^{n+1} \sum_{z \in \Sigma^{n+t}} \sum_{x \in \mathcal{C}} \sum_{x' \in \mathcal{C}} \mathrm{Emb}(z, x) \mathbb{I}(x' \in z) \\ &= \sum_{t=0}^{\infty} (p/2)^t (1-p)^{n+1} \sum_{z \in \Sigma^{n+t}} \left(\sum_{x \in \mathcal{C}} \mathrm{Emb}(z, x) \right) \left(\sum_{x' \in \mathcal{C}} \mathbb{I}(x' \in z) \right). \end{aligned}$$

Since $\sum_{x \in \mathcal{C}} \mathrm{Emb}(z, x)$ and $\sum_{x' \in \mathcal{C}} \mathbb{I}(x' \in z)$ yield the quantities $D^*(z; \mathcal{C})$ and $D(z; \mathcal{C})$, respectively, we obtain (26). ■

Next, similar to Section IV, we look at the quantities $D(z; \mathcal{C})$ and $D^*(z; \mathcal{C})$ for $\mathcal{A}_n$, $\mathcal{E}_n$ and $\mathcal{C}_{n,w}$ as defined in the previous section. Now, the quantity $D(z; \mathcal{A}_n)$ has been studied in other contexts [15]. However, unlike the insertion ball, the quantity $D(z; \{0,1\}^n)$ closely depends on the actual string z. Nevertheless, since we are only interested in the quantity $\sum_{z \in \Sigma^{n+t}} D(z; \mathcal{C}) D^*(z; \mathcal{C})$, we are able to obtain closed expressions for these sums. Using standard techniques in enumerative combinatorics [40], we have the following results.

Proposition 12: If $|z| = n+t$ and $\mathrm{wt}(z) = u$, then

$$D^*(z; \mathcal{A}_n) = \binom{n+t}{t}, \quad (27)$$

$$D^*(z; \mathcal{C}_{n,w}) = \binom{n+t-u}{n-w}\binom{u}{w}, \quad (28)$$

$$D^*(z; \mathcal{E}_n) = \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n+t-u}{n-2k}\binom{u}{2k}. \quad (29)$$

Proof: First, we give a proof of (27). Since $|z| = n+t$, after t deletions, we have a subsequence of length n. Counting multiplicity, the multiset of subsequences of z of length n can be obtained by choosing t positions out of $n+t$ positions of z to delete. Thus the size of the multiset of subsequences is $\binom{n+t}{t}$.

Next, we give a proof of (28). Note that we want a subsequence of length n and weight w, while z is of length $n+t$ and weight u. Therefore, counting multiplicity, the multiset of subsequences of length n and weight w of z can be obtained by choosing w 1's from z and choosing $n-w$ 0's from z. Thus the size of the multiset of subsequences is $\binom{n+t-u}{n-w}\binom{u}{w}$.

Finally, we give a proof of (29). We make use of the formula for $D^*(z; \mathcal{C}_{n,w})$ that we have already obtained, namely $D^*(z; \mathcal{E}_n) = \sum_{k=0}^{\lfloor n/2 \rfloor} D^*(z; \mathcal{C}_{n,2k})$. ■

Using (26–28), we obtain the following closed formulae for the expected number of edges.

Theorem 3: For all n,

$$B(\mathcal{A}_n, \mathrm{Ins}(p)) = 2^n \sum_{t=0}^{\infty} \sum_{i=0}^{t} \binom{n+t}{i}\binom{n+t}{t} (p/2)^t (1-p)^{n+1}. \quad (30)$$

For $w \leq n$,

$$B(\mathcal{C}_{n,w}, \mathrm{Ins}(p)) = \sum_{t=0}^{\infty} \sum_{u \geq 0} \binom{n+t-u}{n-w}\binom{u}{w}\binom{n}{w} I_C(n+t, u, t, w)(p/2)^t (1-p)^{n+1}. \quad (31)$$

[5]We point out a key difference from Proposition 9. Here, $D(z; \mathcal{C})$ denote the number codewords that are *subsequences* of z, while $I(z; \mathcal{C})$ denote the number codewords that are *supersequences* of z. Similar differences are true for D^* and I^*.

Proof: From (26), (27) and finally (6), we have

$$\begin{aligned}
&B(\mathcal{A}_n, \mathrm{Ins}(p))\\
&=\sum_{t=0}^{\infty}\sum_{z\in\Sigma^{n+t}} D(z;\mathcal{A}_n)\binom{n+t}{t}(p/2)^t(1-p)^{n+1}\\
&=\sum_{t=0}^{\infty}(p/2)^t(1-p)^{n+1}\binom{n+t}{t}\sum_{z\in\Sigma^{n+t}}D(z;\mathcal{A}_n)\\
&=\sum_{t=0}^{\infty}(p/2)^t(1-p)^{n+1}\binom{n+t}{t}\sum_{z\in\Sigma^{n}}I(z;\mathcal{A}_{n+t})\\
&=\sum_{t=0}^{\infty}(p/2)^t(1-p)^{n+1}\binom{n+t}{t}2^n\sum_{i=0}^{t}\binom{n+t}{i},
\end{aligned}$$

which results in the expression in (30).

For $w \le n$, applying (26), (28) and finally (7), we have that

$$\begin{aligned}
&B\big(\mathcal{C}_{n,w}, \mathrm{Ins}(p)\big)\\
&=\sum_{t=0}^{\infty}\sum_{z\in\Sigma^{n+t}} D\big(z;\mathcal{C}_{n,w}\big)D^*\big(z;\mathcal{C}_{n,w}\big)(p/2)^t(1-p)^{n+1}\\
&=\sum_{t=0}^{\infty}\sum_{u\ge 0}\sum_{z\in\mathcal{C}_{n+t,u}} D\big(z;\mathcal{C}_{n,w}\big)D^*\big(z;\mathcal{C}_{n,w}\big)(p/2)^t(1-p)^{n+1}\\
&=\sum_{t=0}^{\infty}\sum_{u\ge 0}\binom{n+t-u}{n-w}\binom{u}{w}(p/2)^t(1-p)^{n+1}\sum_{z\in\mathcal{C}_{n+t,u}}D\big(z;\mathcal{C}_{n,w}\big)\\
&=\sum_{t=0}^{\infty}\sum_{u\ge 0}\binom{n+t-u}{n-w}\binom{u}{w}(p/2)^t(1-p)^{n+1}\sum_{z\in\mathcal{C}_{n,w}}I\big(z;\mathcal{C}_{n+t,u}\big)\\
&=\sum_{t=0}^{\infty}\sum_{u\ge 0}\binom{n+t-u}{n-w}\binom{u}{w}\binom{n}{w}(p/2)^t(1-p)^{n+1}I_C(n+t,u,t,w),
\end{aligned}$$

which results in the expression in (31). ■

For the remaining of this section, we demonstrate the polarization behavior of the edge density for the insertion channel. As with Section IV-A, we first show that the edge density approaches zero whenever the insertion probability is strictly less than half.

Proposition 13: Let $\mathcal{C} = \mathcal{A}_n$ and $M = 2^n$. If $0 \le p < 1/2$, then $B(\mathcal{C}, \mathrm{Ins}(p)) = o(M^2)$.

Proof: Let $t' = \left\lceil \frac{np+p-1}{1-p} \right\rceil$ for $p < 1/2$. Observe that $p < 1-p$, and thus, $n\frac{p}{1-p} - 1 < n-1$. Therefore $t' = \left\lceil \frac{np}{1-p} - 1 \right\rceil < n$. Next, we choose a constant $\alpha < 1$ that satisfies both

$$\alpha > \frac{p}{1-p} \quad \text{and} \tag{32}$$

$$\alpha > \frac{\log_2 2(1-p)}{\log_2 \frac{1}{2p}}. \tag{33}$$

Note that α is only dependent on p, and from (32), we have that

$$\alpha n > \frac{np}{1-p} > \left\lceil \frac{np}{1-p} - 1 \right\rceil = t'. \tag{34}$$

Moreover, $2(1-p) < \frac{1}{2p}$ and therefore, $0 < \frac{\log_2 2(1-p)}{\log_2 \frac{1}{2p}} < 1$. Thus, $\alpha < 1$ can be chosen to satisfy (33).

From (30), we have that

$$B(\mathcal{A}_n, \mathrm{Ins}(p)) = 2^n(1-p)^{n+1}\sum_{t=0}^{\infty}\binom{n+t}{t}(p/2)^t\sum_{i=0}^{t}\binom{n+t}{i}.$$

We then split the outer summation of t into two parts, namely from $t = 0$ to $t = \lfloor \alpha n \rfloor$, and then from $t = \lfloor \alpha n \rfloor + 1$ to $t = \infty$.

Observe that

$$\begin{aligned}
&2^n(1-p)^{n+1}\sum_{t=0}^{\lfloor\alpha n\rfloor}\binom{n+t}{t}(p/2)^t\sum_{i=0}^{t}\binom{n+t}{i}\\
&\le 2^n(1-p)^{n+1}\sum_{t=0}^{\lfloor\alpha n\rfloor}\binom{n+t}{t}(p/2)^t 2^{(n+t)H\left(\frac{t}{n+t}\right)}\\
&\le 2^n(1-p)^{n+1}\sum_{t=0}^{\lfloor\alpha n\rfloor}\binom{n+t}{t}p^t 2^{nH\left(\frac{t}{n+t}\right)}\\
&\le 2^n(1-p)^{n+1}2^{nH\left(\frac{\alpha}{1+\alpha}\right)}\sum_{t=0}^{\lfloor\alpha n\rfloor}\binom{n+t}{t}p^t\\
&\le 2^n(1-p)^{n+1}2^{nH\left(\frac{\alpha}{1+\alpha}\right)}\sum_{t=0}^{\infty}\binom{n+t}{t}p^t\\
&\le 2^n 2^{nH\left(\frac{\alpha}{1+\alpha}\right)}.
\end{aligned}$$

Therefore we have

$$\begin{aligned}
&\lim_{n\to\infty}\frac{2^n(1-p)^{n+1}\sum_{t=0}^{\lfloor\alpha n\rfloor}\binom{n+t}{t}(p/2)^t\sum_{i=0}^{t}\binom{n+t}{i}}{2^{2n}}\\
&\le \lim_{n\to\infty}\frac{2^{nH\left(\frac{\alpha}{1+\alpha}\right)}}{2^n} = 0.
\end{aligned} \tag{35}$$

Next, we observe that

$$\begin{aligned}
&2^n(1-p)^{n+1}\sum_{t=\lfloor\alpha n\rfloor+1}^{\infty}\binom{n+t}{t}(p/2)^t\sum_{i=0}^{t}\binom{n+t}{i}\\
&\le 2^n(1-p)^{n+1}\sum_{t=\lfloor\alpha n\rfloor+1}^{\infty}\binom{n+t}{t}(p/2)^t 2^{n+t}\\
&= 2^{2n}(1-p)^{n+1}\sum_{t=\lfloor\alpha n\rfloor+1}^{\infty}\binom{n+t}{t}p^t.
\end{aligned} \tag{36}$$

Let $f(t) = \binom{n+t}{t}p^t$. Observe that $f(t+1)/f(t) = \frac{\binom{n+t+1}{t+1}p^{t+1}}{\binom{n+t}{t}p^t} = \frac{n+t+1}{t+1}p < 1$ if and only if $t > \frac{np+p-1}{1-p}$. Therefore, $f(t)$ achieves its maximum value at $t = t'$. Furthermore, $f(t+1) < f(t)$ if $t \ge t'$. Lastly, from (34), we have $\lfloor \alpha n \rfloor + 1 \ge \alpha n \ge t'$. All these imply that we can upper bound $\sum_{t=\lfloor\alpha n\rfloor+1}^{\infty}\binom{n+t}{t}p^t$ as a geometric series with initial value $\binom{n+\lfloor\alpha n\rfloor+1}{\lfloor\alpha n\rfloor+1}p^{\alpha n}$ and ratio $\frac{n+t+1}{t+1}p = (\frac{n}{t+1}+1)p \le (\frac{1}{\alpha}+1)p$ which is less than 1 because of condition (32). Our observation can be summarized as follows

$$\sum_{t=\lfloor\alpha n\rfloor+1}^{\infty}\binom{n+t}{t}p^t \le \frac{\binom{n+\lfloor\alpha n\rfloor+1}{\lfloor\alpha n\rfloor+1}p^{\lfloor\alpha n\rfloor+1}}{1-\left(\frac{1}{\alpha}+1\right)p} \le \frac{2^{n+\lfloor\alpha n\rfloor+1}p^{\lfloor\alpha n\rfloor+1}}{1-\left(\frac{1}{\alpha}+1\right)p}. \tag{37}$$

Thus combining (36) and (37), we have

$$\lim_{n\to\infty}\frac{2^n(1-p)^{n+1}\sum_{t=\lfloor\alpha n\rfloor+1}^{\infty}\binom{n+t}{t}(p/2)^t\sum_{i=0}^{t}\binom{n+t}{i}}{2^{2n}}$$

$$\leq \lim_{n\to\infty} \frac{1-p}{1-\left(\frac{1}{\alpha}+1\right)p}(2(1-p))^n(2p)^{\lfloor \alpha n\rfloor+1}$$

$$\leq 2p\frac{1-p}{1-\left(\frac{1}{\alpha}+1\right)p}\lim_{n\to\infty}\left(\frac{2(1-p)}{\left(\frac{1}{2p}\right)^{\alpha}}\right)^n = 0, \tag{38}$$

where the last equation comes from condition (33).

Finally, combining (35) and (38), we have that $2^n(1-p)^{n+1}\sum_{t=0}^{\infty}\binom{n+t}{t}(p/2)^t\sum_{i=0}^{t}\binom{n+t}{i} = o(2^{2n})$. ■

To conclude this section, we have the following analogue of Proposition 9.

Proposition 14:

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

Proof:

- When $p < 1/2$, the limit value is immediate from Proposition 13.
- When $p > 1/2$, we want to show that $\lim_{n\to\infty}\frac{2^{2n}-B(\mathcal{A}_n,\mathrm{Ins}(p))}{2^{2n}} = 0$. Note that

$$\begin{aligned}
&2^n(1-p)^{n+1}\sum_{t=0}^{\infty}\binom{n+t}{t}(p/2)^t\sum_{i=0}^{n+t}\binom{n+t}{i}\\
&= 2^n(1-p)^{n+1}\sum_{t=0}^{\infty}\binom{n+t}{t}(p/2)^t2^{n+t}\\
&= 2^{2n}(1-p)^{n+1}\sum_{t=0}^{\infty}\binom{n+t}{t}p^t\\
&= 2^{2n}(1-p)^{n+1}\frac{1}{(1-p)^{n+1}} = 2^{2n}.
\end{aligned}\tag{39}$$

Therefore from (30) and (39), we have

$$\begin{aligned}
&2^{2n} - B(\mathcal{A}_n, \mathrm{Ins}(p))\\
&= 2^n(1-p)^{n+1}\sum_{t=0}^{\infty}\binom{n+t}{t}(p/2)^t\sum_{i=t+1}^{n+t}\binom{n+t}{i}\\
&= 2^n(1-p)^{n+1}\sum_{t=0}^{\infty}\binom{n+t}{t}(p/2)^t\sum_{i=0}^{n-1}\binom{n+t}{i}.
\end{aligned}$$

First, we choose a constant $\alpha > 1$ such that

$$\alpha < \frac{\log_2\frac{1}{2(1-p)}}{\log_2 2p} \text{ and} \tag{40}$$

$$\alpha < \frac{p}{1-p}. \tag{41}$$

This is feasible, because $\frac{1}{2(1-p)} > 2p$ for $1/2 < p < 1$, and thus $\log_2\frac{1}{2(1-p)} > \log_2 2p$. Observe also that $f(t) = \binom{n+t}{t}p^t$ is maximized when $t = t' = \left\lceil\frac{np}{1-p} - 1\right\rceil \geq n$. Furthermore, $f(t+1) > f(t)$ if $t \leq \alpha n \leq t'$. Moreover, from (40), it implies that $(2p)^\alpha < \frac{1}{2(1-p)}$, and thus

$$2(1-p)(2p)^\alpha \triangleq \beta < 1. \tag{42}$$

Now, similar to Proposition 13, we split the summation of t into two parts. Using all these information, we have

$$\begin{aligned}
&2^n(1-p)^{n+1}\sum_{t=0}^{\alpha n}\binom{n+t}{t}(p/2)^t\sum_{i=0}^{n-1}\binom{n+t}{i}\\
&\leq 2^n(1-p)^{n+1}\sum_{t=0}^{\alpha n}\binom{n+t}{t}(p/2)^t2^{n+t}\\
&= 2^{2n}(1-p)^{n+1}\sum_{t=0}^{\alpha n}\binom{n+t}{t}p^t\\
&\leq 2^{2n}(1-p)^{n+1}\alpha n\binom{n+\alpha n}{\alpha n}p^{\alpha n}\\
&\leq 2^{2n}(1-p)^{n+1}\alpha n2^{n+\alpha n}p^{\alpha n}\\
&= 2^{2n}\alpha(1-p)n\big(2(1-p)(2p)^\alpha\big)^n\\
&= 2^{2n}\alpha(1-p)n\beta^n.
\end{aligned}\tag{43}$$

Secondly, for the second part of the summation, we have

$$\begin{aligned}
&2^n(1-p)^{n+1}\sum_{t=\alpha n}^{\infty}\binom{n+t}{t}(p/2)^t\sum_{i=0}^{n-1}\binom{n+t}{i}\\
&\leq 2^n(1-p)^{n+1}\sum_{t=\alpha n}^{\infty}\binom{n+t}{t}(p/2)^t2^{(n+t)H\left(\frac{n-1}{n+t}\right)}\\
&\leq 2^n(1-p)^{n+1}\sum_{t=\alpha n}^{\infty}\binom{n+t}{t}(p/2)^t2^{(n+t)H\left(\frac{n}{n+\alpha n}\right)}\\
&\leq 2^n(1-p)^{n+1}2^{nH\left(\frac{1}{1+\alpha}\right)}\sum_{t=\alpha n}^{\infty}\binom{n+t}{t}p^t\\
&\leq 2^n(1-p)^{n+1}2^{nH\left(\frac{1}{1+\alpha}\right)}\frac{1}{(1-p)^{n+1}}\\
&= 2^n2^{nH\left(\frac{1}{1+\alpha}\right)}.
\end{aligned}\tag{44}$$

Here, the penultimate inequality follows from the negative binomial series expansion: $(1-p)^{-n-1} = \sum_{t\geq 0}\binom{n+t}{t}p^t$. Combining (43) and (44), we have

$$\begin{aligned}
&\lim_{n\to\infty}\frac{2^{2n}-B(\mathcal{A}_n,\mathrm{Ins}(p))}{2^{2n}}\\
&\leq \lim_{n\to\infty}\frac{2^{2n}\alpha(1-p)n\beta^n + 2^n2^{nH\left(\frac{1}{1+\alpha}\right)}}{2^{2n}}\\
&= \lim_{n\to\infty}\alpha(1-p)n\beta^n + \lim_{n\to\infty}\frac{2^{nH\left(\frac{1}{1+\alpha}\right)}}{2^n}\\
&= 0 + 0,
\end{aligned}$$

where the last equation holds because $\beta < 1$ from (42) and $H(\frac{1}{1+\alpha}) < 1$, since $\alpha > 1$. ■

As before, we exhibit the polarization behavior numerically for the insertion channel in Figure 1(c). Also, we make the following conjecture on the asymptotic behavior with $p = 1/2$.

Conjecture 2: Let $p = 1/2$. Then

$$\lim_{n\to\infty}\frac{B(\mathcal{A}_n,\mathrm{Ins}(p))}{|\mathcal{A}_n|^2} = \frac{1}{2}.$$

VI. Concluding Remarks

We conclude by discussing extensions and future work.

- *Expected edge density for other codebooks:* In this paper, we provided closed formulae and efficient methods to compute the expected edge density for some code families. In particular, when the codebook comprises all binary words, we show that this quantity is sub-quadratic when the deletion or insertion probability is less than 1/2. In contrast, for the binary erasure channel, in [19], this quantity was shown to be linear for certain families of code. Therefore, it will be of interest to find such code families for the insertion and deletion channels.
- *Polarization behavior of edge density:* In Sections IV-A and V, we demonstrated that the expected edge density for $\mathcal{A}_n$ polarizes for both the deletion and insertion channels. Nevertheless, we also observe similar behavior for constant weight codebooks and hence, we have Conjecture 1. Again, it will be interesting to exhibit this polarization behavior for other code families and determine the corresponding probability threshold.
- *Data-driven decoder:* In Section II-D, we described a simple peeling decoder and in Corollary 1, we provided a rudimentary analysis of the decoder by estimating the number of degree-one nodes. In our future work, we refine this analysis and provide sharper estimates on the probability of successful decoding.

  Recall that we are motivated by applications in DNA-based data storage, where we are required to identify files using their reads. In our approach, we ignored the data blocks contents and only made use the noisy reads of the addresses. In our preliminary studies [41], we propose a method to efficiently use this data to increase the identification.

Acknowledgment

Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] J. Chrisnata, H. M. Kiah, A. Vardy, and E. Yaakobi. "Bee identification problem for DNA strands," in *Proc. IEEE Int. Symp. Inf. Theory*, 2022, pp. 969–974.

[2] J. D. Watson and F. H. Crick, "Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid," *Nature*, vol. 171, no. 4356, pp. 737–738, 1953.

[3] J. D. Watson and F. H. Crick, "The structure of DNA," in *Cold Spring Harbor Symposia on Quantitative Biology*, vol. 18. Cold Spring Harbor, NY, USA: Cold Spring Harbor Lab. Press Jan. 1953, pp. 123–131.

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

[5] N. Goldman et al., "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 7435, pp. 77–80, 2013.

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

[7] O. Milenkovic, R. Gabrys, H. M. Kiah, and S. H. T. Yazdi, "Exabytes in a test tube," *IEEE Spectr.*, vol. 55, no. 5, pp. 40–45, May 2018.

[8] I. Shomorony and R. Heckel, "Information-theoretic foundations of DNA data storage," *Found. Trends® Commun. Inf. Theory*, vol. 19, no. 1, pp. 1–106, 2022

[9] L. Organick et al., "Random access in large-scale DNA data storage," *Nat. Biotechnol.*, vol. 36, no. 3, pp242–248, 2018.

[10] J. L. Schmid-Burgk et al., "LAMP-Seq: Population-scale COVID-19 diagnostics using combinatorial barcoding," 2020, *bioRxiv:2020.04.06.025635*.

[11] J. Li et al., "Rapid detection of SARS-CoV-2 and other respiratory viruses by using LAMP method with Nanopore Flongle workflow," 2020, *bioRxiv:2020.06.03.131474*.

[12] P. James et al., "LamPORE: Rapid, accurate and highly scalable molecular screening for SARS-CoV-2 infection, based on nanopore sequencing," 2020, *medRxiv:2020.08.07.20161737*.

[13] L. Peto et al., "Diagnosis of SARS-CoV-2 infection with LamPORE, a high-throughput platform combining loop-mediated isothermal amplification and nanopore sequencing," 2020, *medRxiv:2020.09.18.20195370*.

[14] A. S. Booeshaghi et al., "Reliable and accurate diagnostics from highly multiplexed sequencing assays," *Sci. Rep.*, vol. 10, Dec. 2020, Art. no. 21759.

[15] V. I. Levenshtein, "Efficient reconstruction of sequences," *IEEE Trans. Inf. Theory*, vol. 47, no. 1, pp. 2–22, Jan. 2001.

[16] K. Cai, H. M. Kiah, T. T. Nguyen, and E. Yaakobi, "Coding for sequence reconstruction for single edits," *IEEE Trans. Inf. Theory*, vol. 68, no. 1, pp. 66–79, Jan. 2022.

[17] C. Rashtchian et al., "Clustering billions of reads for DNA data storage," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 3360–3371.

[18] 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, Nov. 2019.

[19] H. M. Kiah, A. Vardy, and H. Yaoz, "Efficient Algorithms for the Bee-Identification Problem," *IEEE J. Sel. Areas Inf. Theory*, early access, Jul. 2023, doi: 10.1109/JSAIT.2023.3296077.

[20] E. A. Ratzner, "Marker codes for channels with insertions and deletions," *Annales Des Télécommun.*, vol. 60, pp. 29–44, Feb. 2005. [Online]. Available: https://doi.org/10.1007/BF03219806

[21] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *J. ACM*, vol. 19, no. 2, pp. 248–264, 1972.

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

[23] T. Shinkar, E. Yaakobi, A. Lenz, and A. Wachter-Zeh, "Clustering-correcting codes," *IEEE Trans. Inf. Theory*, vol. 68, no. 3, pp. 1560–1580, Mar. 2022. doi: 10.1109/TIT.2021.3127174.

[24] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding over sets for DNA storage," *IEEE Trans. Inf. Theory*, vol. 66, no. 4, pp. 2331–2351, Apr. 2020.

[25] J. Sima, N. Raviv, and J. Bruck, "On coding over sliced information," *IEEE Trans. Inf. Theory*, vol. 67, no. 5, pp. 2793–2807, May 2021.

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

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

[28] 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.

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

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

[31] S. Yazdi, H. M. Kiah, R. Gabrys, and O. Milenkovic. "Mutually uncorrelated primers for DNA-based data storage," *IEEE Trans. Inf. Theory*, vol. 64, no. 9, pp. 6283–6296, Sep. 2018.

[32] M. Levy and E. Yaakobi, "Mutually uncorrelated codes for DNA storage," *IEEE Trans. Inf. Theory*, vol. 65, no. 6, pp. 3671–3691, Jun. 2019.

[33] Y. M. Chee, H. M. Kiah, and H. Wei, "Efficient and explicit balanced primer codes," *IEEE Trans. Inf. Theory*, vol. 66, no. 9, pp. 5344–5357, Sep. 2020.

[34] S. R. Srinivasavaradhan, S. Gopi, H. D. Pfister, and S. Yekhanin, "Trellis BMA: Coded trace reconstruction on IDS channels for DNA storage," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2021, pp. 2453–2458, doi: 10.1109/ISIT45174.2021.9517821.

[35] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversal," *English Transl. Soviet Phys. Dokl.*, vol. 10, no. 8, pp. 707–710, 1966.

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

[37] A. Vardy, "Algorithmic complexity in coding theory and the minimum distance problem," in *Proc. 29th Annu. ACM Symp. Theory Comput.*, May 1997, pp. 92–109.

[38] V. I. Levenshtein, "Elements of coding theory," in *Diskretnaya Matematika I Matematicheskie Voprosy Kibernetiki*. Berlin, Germany: Academie-Verlag, 1974, pp. 207–305.

[39] A. Atashpendar et al., "From clustering supersequences to entropy minimizing subsequences for single and double deletions," 2018, *arXiv:1802.00703*.

[40] P. Flajolet and R. Sedgewick, *Analytic Combinatorics*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[41] S. Singhvi, A. Boruchovsky, H. M. Kiah, and E. Yaakobi. "Data-Driven Bee Identification for DNA Strands," in *Proc. IEEE Int. Symp. Inf. Theory*, 2023, pp. 797–852.

Johan Chrisnata received the bachelor's degree in mathematics from Nanyang Technological University (NTU), Singapore, in 2015, and the joint Ph.D. degree in mathematics from the School of Physical and Mathematical Sciences, NTU and computer science from the Department of Computer Science, Technion University, Israel. From August 2015 until August 2018, he was a Research Officer with NTU, where he is currently a Research Fellow. His research interest includes enumerative combinatorics and coding theory, in particular DNA-based data storage and sequences.

Han Mao Kiah (Senior Member, IEEE) received the Ph.D. degree in mathematics from Nanyang Technological University (NTU), Singapore, in 2014. From 2014 to 2015, he was a Postdoctoral Research Associate with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign. From 2015 to 2018, he was a Lecturer with the School of Physical and Mathematical Sciences, NTU, where he is currently an Assistant Professor. His research interests include DNA-based data storage, coding theory, enumerative combinatorics, and combinatorial design theory.

Alexander Vardy was born in Moscow, in 1963. He received the B.Sc. (summa cum laude) degree from Technion, Israel, in 1985, and the Ph.D. degree from Tel-Aviv University, Israel, in 1991.

From 1985 to 1990, he was with Israeli Air Force, where he worked on electronic counter measures systems and algorithms. From 1992 to 1993, he was a Visiting Scientist with IBM Almaden Research Center, San Jose, CA, USA. From 1993 to 1998, he was with the University of Illinois at Urbana-Champaign, first as an Assistant Professor and then as an Associate Professor. From 1998 to 2022, he was the Jack Keil Wolf Chair Professor with the Department of Electrical and Computer Engineering and the Department of Computer Science, University of California San Diego (UCSD). While on sabbatical with UCSD, he has held long-term visiting appointments with CNRS, France, EPFL, Switzerland, the Technion-Israel Institute of Technology, and Nanyang Technological University, Singapore. His research interests include error-correcting codes, algebraic and iterative decoding algorithms, lattices and sphere packings, coding for storage systems, cryptography, computational complexity theory, and fun math problems. He received the IBM Invention Achievement Award in 1993 and the NSF Research Initiation and CAREER Awards in 1994 and 1995, respectively. In 1996, he was appointed as a Fellow of the Center for Advanced Study, University of Illinois, and received the Xerox Award for Faculty Research. He received the IEEE Information Theory Society Paper Award (jointly with Ralf Koetter) in 2004. In 2005, he received the Fulbright Senior Scholar Fellowship and the Best Paper Award at the IEEE Symposium on Foundations of Computer Science. In 2017, his work on polar codes was recognized by the IEEE Communications and Information Theory Societies Joint Paper Award. From 1995 to 1998, he was an Associate Editor for Coding Theory. From 1998 to 2001, he was the Editor-in-Chief of the IEEE TRANSACTIONS ON INFORMATION THEORY. He has been a member of the Board of Governors of the IEEE Information Theory Society from 1998 to 2006 and from 2011 to 2017. In 1996, he became a Fellow of the David and Lucile Packard Foundation.

Eitan Yaakobi (Senior Member, IEEE) received the B.A. degree in computer science and mathematics and the M.Sc. degree in computer science from the Technion—Israel Institute of Technology, Haifa, Israel, in 2005 and 2007, respectively, and the Ph.D. degree in electrical engineering from the University of California at San Diego, San Diego, in 2011.

He is an Associate Professor with the Computer Science Department, Technion—Israel Institute of Technology. He also holds a courtesy appointment with the Technion's Electrical and Computer Engineering Department. From 2011 to 2013, he was a Postdoctoral Researcher with the Department of Electrical Engineering, California Institute of Technology and the Center for Memory and Recording Research, University of California at San Diego, San Diego. Since 2016, he has been with the Center for Memory and Recording Research, University of California San Diego, La Jolla, and since 2018, he has been with the Institute of Advanced Studies, Technical University of Munich, where he holds a four-year Hans Fischer Fellowship, funded by the German Excellence Initiative and the EU 7th Framework Program. His research interests include information and coding theory with applications to non-volatile memories, associative memories, DNA storage, data storage and retrieval, and private information retrieval. He received the Marconi Society Young Scholar in 2009 and the Intel Ph.D. Fellowship from 2010 to 2011. Since 2020, he has been serving as an Associate Editor for Coding and Decoding for the IEEE TRANSACTIONS ON INFORMATION THEORY. He is a recipient of several grants, including the ERC Consolidator Grant.