Theory and Practice of DNA Storage

Daniella Bar-Lev

Theory and Practice of DNA Storage

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Daniella Bar-Lev

Submitted to the Senate of
the Technion – Israel Institute of Technology
Sivan 5784 Haifa July 2024

The research thesis was done under the supervision of Prof. Tuvi Etzion and Prof. Eitan Yaakobi in the Computer Science Department.

The author of this thesis states that the research, including the collection, processing, and presentation of data, addressing and comparing to previous research, etc., was done entirely in an honest way, as expected from scientific research that is conducted according to the ethical standards of the academic world. Also, reporting the research and its results in this thesis was done in an honest and complete manner, according to the same standards.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisors, Prof. Tuvi Etzion and Prof. Eitan Yaakobi. Choosing to work with both of you has undoubtedly been one of the best decisions I have ever made. I am immensely grateful for the opportunity to learn and grow under your guidance.

To Tuvi, thank you for your constant encouragement and for teaching me what it means to be a good researcher. Your lessons extended far beyond academic texts, guiding me on how to conduct myself in the academic world. You believed in me even when I doubted myself, instilling confidence that has been instrumental throughout this journey. Your endless patience, your willingness to help with any matter, and your thoughtful advice—always with my best interest at heart—are qualities I deeply appreciate. To Eitan, you have been an incredible role model, continually inspiring me to push beyond my limits. Your enthusiasm and intellectual curiosity have motivated me to explore new ideas, no matter how unconventional they seem. Thank you for fostering my curiosity and always supporting my boldest ideas. Both of you opened countless doors for me and provided opportunities I could only dream of. Your unwavering support—from the very first steps of my academic journey to the point of independence in research—has shaped me into the researcher I am today. I am forever grateful for the scholarships, the opportunities, and most importantly, the knowledge you imparted. I will do my best to apply these lessons throughout my career, and I hope to be able to give others even a portion of everything I have received from you.

A heartfelt thank you goes to Omer Sabary. You have been an essential part of this journey, making it not only academically fulfilling but also genuinely enjoyable and exciting. I could always rely on you, as both, a professional collaborator and a close friend. Whether celebrating successes or sharing fears and dreams, your support and our collaboration on numerous projects have made this experience deeply enriching. To Dr. Itai Orr, I am truly grateful for the countless discussions and exchange of ideas we've shared. Your experience and knowledge have introduced me to new concepts and perspectives, and every learning moment has been a pleasure. But beyond that, I deeply value your friendship. It has been a source of great support and joy throughout this journey. Working with both of you has been a privilege, and I'm grateful for everything we've accomplished together.

I would also like to extend my thanks to all my colleagues and collaborators during my PhD. As Terry Pratchett once said, "The best research you can do is talk to people." Each of you has contributed to my journey in a meaningful way. I have learned from and enjoyed working with all of you. This experience would not have been the same without your interest, support, and companionship. Whether through advice, direction, or even casual conversations

over lunch or coffee breaks, I am thankful for every moment we shared, from times of difficulty to times of celebration. In particular, I would like to acknowledge Prof. Zohar Yakhini, Dr. Ryan Gabrys, Dr. Yonatan Yehezkeally, Prof. Ori Rottenstreich, Inbal Preuss, Dganit Hanania, Avital Boruchovsky, Orian Leitersdorf, Adir Kobovich, and Anina Gruica, for their significant contributions, support, and friendship throughout this journey.

Special thanks go to my thesis examiners, Prof. Ronny Roth and Prof. Moshe Schwartz, for their thorough reading of my work and for their insightful comments. Your feedback greatly contributed to improving the quality of this thesis, and I truly appreciate your time and effort. Moreover, I would like to extend my gratitude for your presence throughout this journey. To Ronny, thank you for always attending my seminars and talks, showing genuine interest in my research, and offering thoughtful ideas and suggestions. You continuously challenged me to refine my work, sharpen my skills, and grow as both a researcher and educator. To Moshe, I am deeply grateful for always being available for consultation and advice, especially regarding my academic aspirations.

I also want to express my appreciation to the staff at the Computer Science Department for their constant support. In particular, I would like to thank Yifat Chen-Solomon for always assisting me throughout my studies when needed.

Finally, I would like to dedicate this thesis to my family. To my husband, Shahal Weissleder, who has been by my side throughout this entire journey—thank you for your unwavering support, encouragement, and genuine interest in my research. Your patience and understanding gave me the space and time to work, and your strength became the foundation on which I could build my own. You were, and continue to be, my pillar of strength and the source of my determination. To my parents, Lea and Albert Bar-Lev, my sister Eden Bar-Lev, and my grandparents, Tamara and Yaakov Eligulashvili, and Ziva Daniel —thank you for raising me with the tools, curiosity, and soft skills that made this journey possible. You have always encouraged my inquisitive nature and provided me with everything I needed to succeed. More importantly, your constant love, care, and support, through every challenge life has brought, have shaped me into the person I am today. Your belief in me, even from a young age, has been a constant source of motivation. I am forever grateful for your presence in my life. Last, but not least, a special thank you to my cat, Steven, for bringing immeasurable joy into my life. From sitting by my side during countless Zoom meetings to keeping me company through every tutorial and lecture I presented online, you've been an unexpected yet delightful companion throughout this process.

The generous financial help of Miriam and Aaron Gutwirth, the Israel Science Foundation, and the Henry and Marilyn Taub Faculty of Computer Science at the Technion is gratefully acknowledged.

Publications List - Daniella Bar-Lev

Journals:

Published Papers

[1] **D. Bar-Lev**, O. Sabary, and E. Yaakobi. "Exciting Coding Problems for DNA-Based Storage Systems," *Notices of the American Mathematical Society*, December 2022.

[2] **D. Bar-Lev**, T. Etzion, and E. Yaakobi. "On the Size of Balls and Anticodes of Small Diameter under the Fixed-Length Levenshtein Metric," *IEEE Transactions on Information Theory*, vol. 69, no. 4, pp. 2324-2340, April 2023.

[3] Y. Yehezkeally, **D. Bar-Lev**, S. Marcovich, and E. Yaakobi. "Generalized Unique Reconstruction from Substrings," *IEEE Transactions on Information Theory*, vol. 69, no. 9, pp. 5648-5659, September 2023.

[4] **D. Bar-Lev**, S. Marcovich, E. Yaakobi, and Y. Yehezkeally. "Adversarial Torn-Paper Codes," *IEEE Transactions on Information Theory*, vol. 69, no. 10, pp. 6414-6427, October 2023.

[5] Y. Nogin, **D. Bar-Lev**, D. Hanania, T. Detinis Zur, Y. Ebenstein, E. Yaakobi, N. Weinberger, and Y. Shechtman. "Design of optimal labeling patterns for optical genome mapping via information theory," *Bioinformatics*, vol. 39, no. 10, October 2023.

Submitted Papers

[6] **D. Bar-Lev**, O. Sabary, R. Gabrys, and E. Yaakobi. "Cover Your Bases: How to Minimize the Sequencing Coverage in DNA Storage Systems," submitted to *IEEE Transactions on Information Theory*.

[7] **D. Bar-Lev**, O. Sabary, and E. Yaakobi. "The Zettabyte Era is in Our DNA," submitted to *Nature Computational Science*.

[8] D. Hanania, **D. Bar-Lev**, Y. Nogin, and E. Yaakobi. "On the Capacity of DNA Labeling," submitted to *IEEE Transactions on Information Theory*.

[9] **D. Bar-Lev**, I. Orr, O. Sabary, T. Etzion, and E. Yaakobi "Deep DNA Storage: Scalable and Robust DNA Storage via Coding Theory and Deep Learning," submitted to *Nature Machine Intelligence*.

[10] A. Boruchovsky, **D. Bar-Lev**, and E. Yaakobi. "DNA-Correcting Codes: End-to-end Correction in DNA Storage Systems," submitted to *IEEE Transactions on Information Theory*.

In Preparation (for a journal submission)

[11] **D. Bar-Lev**, A. Kobovich, O. Leitersdorf, and E. Yaakobi. "Universal Framework for Parametric Constrained Coding," *arXiv preprint*.

Peer-Reviewed Conference Proceedings

[12] **D. Bar-Lev**, T. Etzion, and E. Yaakobi. "On Levenshtein Balls with Radius One," *IEEE International Symposium on Information Theory (ISIT)*, Melbourne, Australia, 12-20 July, 2021 (virtual). (Contained in [2]).

[13] **D. Bar-Lev**, Y. Gershon, O. Sabary, and E. Yaakobi. "Decoding for Optimal Expected Normalized Distance over the t-Deletion Channel," *IEEE International Symposium on Information Theory (ISIT)*, Melbourne, Australia, 12-20 July 2021 (virtual).

[14] **D. Bar-Lev**, O. Sabary, Y. Gershon and E. Yaakobi. "The Intersection of Insertion and Deletion Balls," *IEEE Information Theory Workshop (ITW)*, Kanazawa, Japan, 17-21 October, 2021 (virtual).

[15] **D. Bar-Lev**, S. Marcovich, E. Yaakobi, Y. Yehezkeally. "Adversarial Torn-Paper Codes," *IEEE International Symposium on Information Theory (ISIT)*, Espoo, Finland, June 26-July 1, 2022. (Contained in [4]).

[16] Y. Yehezkeally, **D. Bar-Lev**, S. Marcovich, and E. Yaakobi. "Reconstruction from Substrings with Partial Overlap," *IEEE International Symposium on Information Theory and its Applications (ISITA)*, Tsukuba, Japan, 17-19 October, 2022. (Contained in [3])

[17] A. Kobovich, O. Leitersdorf, **D. Bar-Lev**, and E. Yaakobi. "Codes for Constrained Periodicity," *IEEE International Symposium on Information Theory and its Applications (ISITA)*, Tsukuba, Japan, 17-19 October, 2022. **Best Paper Award**.

[18] S. Singhvi, O. Sabary, **D. Bar-Lev**, and E. Yaakobi. "The Input and Output Entropies of the k-Deletion/Insertion Channel with Small Radii," *IEEE Information Theory Workshop (ITW)*, Mumbai, India, 6-9 November, 2022.

[19] **D. Bar-Lev**, O. Sabary, R. Gabrys, and E. Yaakobi. "Cover Your Bases: How to Minimize the Sequencing Coverage in DNA Storage Systems," *IEEE International Symposium on Information Theory (ISIT)*, Taipei, Taiwan, 25-30 June, 2023. (Contained in [6]).

[20] **D. Bar-Lev**, A. Mizrahi, T. Etzion, O. Rottenstreich, and E. Yaakobi. "Codes for IBLTs with Listing Guarantees," *IEEE International Symposium on Information Theory (ISIT)*, Taipei, Taiwan, 25-30 June, 2023.

[21] D. Hanania, **D. Bar-Lev**, Y. Nogin, and E. Yaakobi. "On the Capacity of DNA Labeling," *IEEE International Symposium on Information Theory (ISIT)*, Taipei, Taiwan, 25-30 June, 2023. (Contained in [8])

[22] A. Boruchovsky, **D. Bar-Lev**, and E. Yaakobi. "DNA Correcting Codes: End-to-end Correction in DNA Storage Systems," *IEEE International Symposium on Information Theory (ISIT)*, Taipei, Taiwan, 25-30 June, 2023. (Contained in [10])

[23] A. Mizrahi, **D. Bar-Lev**, E. Yaakobi, and O. Rottenstreich. "Invertible Bloom Look-Up Tables with Listing Guarantees," *ACM SIGMETRICS*, Venice, Italy, 10-14 June, 2024.

[24] A. Gruica, **D. Bar-Lev**, A. Ravagnani, and E. Yaakobi. "Reducing Coverage Depth in DNA Storage: A Combinatorial Perspective on Random Access Efficiency," *IEEE International Symposium on Information Theory (ISIT)*, Athens, Greece, 7-12 July, 2024.

[25] **D. Bar-Lev**, A. Kobovich, O. Leitersdorf, and E. Yaakobi. "Optimal Almost-Balanced Sequences," *IEEE International Symposium on Information Theory (ISIT)*, Athens, Greece, 7-12 July, 2024.

[26] A. Kobovich, O. Leitersdorf, **D. Bar-Lev**, and E. Yaakobi. "Universal Framework for Parametric Constrained Coding," *IEEE International Symposium on Information Theory (ISIT)*, Athens, Greece, 7-12 July, 2024. (Contained in [11]).

[27] **D. Bar-Lev**, T. Etzion, E. Yaakobi, and Z. Yakhini. "Representing Information on DNA using Patterns Induced by Enzymatic Labeling," *IEEE International Symposium on Information Theory (ISIT)*, Athens, Greece, 7-12 July, 2024.

Contents

List of Figures

List of Tables

Abstract

The global data sphere is expanding exponentially, projected to hit 180 Zettabytes by 2025, whereas current technologies are not anticipated to scale at nearly the same rate. DNA-based storage emerges as a crucial solution to this gap, enabling digital information to be archived in DNA molecules (where the DNA letters are A,C,G, and T). This method enjoys major advantages over magnetic and optical storage solutions such as exceptional information density, enhanced data durability, and negligible power consumption to maintain data integrity.

This research delves into intrinsic error characteristics of DNA-based storage systems to devise robust coding strategies and innovative algorithms for enhanced reliability, efficiency, scalability, and cost-effectiveness. This Ph.D. research propels DNA storage feasibility while contributing to the foundation of this theory.

The work analyzes combinatorial structures tied to errors that are common in DNA-based storage systems, like insertions and deletions. In particular, the work offers a meticulous analysis of the sizes of error balls under the Fixed-Length Levenshtein (FLL) metric, which is the metric to consider when the number of insertions is equal to the number of deletions. The size of anticodes under the FLL metric, as well as the size of the intersections between insertion and deletion balls, are also being studied. These explorations illuminate DNA storage's unique error behavior, offering insights into its fundamental limits and capabilities.

Moreover, to advance DNA storage-related coding techniques, this work introduces novel and efficient coding schemes in addressing challenges unique to DNA-based storage systems, such as DNA fragmentation. Moreover, the work delves into the problem of almost-balanced sequences. This problem addresses a fundamental challenge in constrained coding and carries significant implications for the reliability of DNA storage systems. Specifically, maintaining approximately balanced GC content (typically between 45% and 55% GC ratio) within stored sequences emerges as a powerful strategy for error mitigation. These efforts underscore coding techniques' pivotal role in DNA storage advancement.

Bridging theory with application, this work introduces a pioneering proof-of-concept for a scalable DNA-based storage pipeline that integrates Deep Neural Networks with coding strategies. To reduce sequencing costs and latency while maintaining high retrieval accuracy, the DNA coverage depth problem, a delicate balance involving sequencing costs, latency, and retrieval accuracy, is introduced and addressed.

Lastly, this work introduces a novel theoretical framework for representing information using DNA molecules, aiming to bypass the need for DNA synthesis, a slow and expensive process in traditional DNA-based storage methods.

In conclusion, this research enriches DNA-based storage methodologies by addressing inherent error characteristics and proposing innovative coding schemes. The integration of Deep Neural Networks offers scalability, while a novel theoretical framework aims to mitigate the time and costs associated with DNA sequencing and synthesis, marking significant strides toward practical DNA storage solutions.

Abbreviations and Notations

Abbreviations and Notations for Chapter 3

n	—	Sequence (word) length
$\mathbb{Z}_q$	—	The set of integers $\{0, 1, \ldots, q-1\}$
$\mathbb{Z}_q^n$	—	The set of all sequences of length n over the alphabet $\mathbb{Z}_q$
$\mathbb{Z}_q^*$	—	$\bigcup_{n=0}^{\infty} \mathbb{Z}_q^n$
$[n]$	—	The set of integers $\{1, 2, \ldots, n\}$
$\mathrm{wt}(\boldsymbol{x})$	—	The Hamming weight of the sequence $\boldsymbol{x}$
$d_H(\boldsymbol{x}, \boldsymbol{y})$	—	The Hamming distance between two sequences $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^n$
$\mathcal{H}_t(\boldsymbol{x})$	—	The Hamming t-ball centered at $\boldsymbol{x}$
$\mathcal{D}_t(\boldsymbol{x})$	—	The deletion t-sphere centered at $\boldsymbol{x}$
$D_q(n, t)$	—	The size of the largest deletion t-sphere in $\mathbb{Z}_q^n$
$\mathcal{I}_t(\boldsymbol{x})$	—	The insertion t-sphere centered at $\boldsymbol{x}$
$\lvert S \rvert$	—	The cardinality of the set S
$\rho(\boldsymbol{x})$	—	The number of runs in $\boldsymbol{x}$
$d_L(\boldsymbol{x}, \boldsymbol{y})$	—	The Levenshtein distance between two words $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^*$
$d_E(\boldsymbol{x}, \boldsymbol{y})$	—	The edit distance between two words $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^*$
$\widehat{\mathcal{L}}_t(\boldsymbol{x})$	—	The Levenshtein t-ball centered at $\boldsymbol{x}$
$d_\ell(\boldsymbol{x}, \boldsymbol{y})$	—	The Fixed Length Levenshtein (FLL) distance between two words $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^n$
$\mathcal{L}_t(\boldsymbol{x})$	—	The FLL t-ball centered at $\boldsymbol{x}$
$\boldsymbol{x}_{[i,j]}$	—	The substring of $\boldsymbol{x}$ from index i to index j, i.e., $x_i x_{i+1} \cdots x_j$
$A(\boldsymbol{x})$	—	The number of maximal alternating segments of the sequence $\boldsymbol{x}$
s_i	—	The length of the i-th maximal alternating segment of $\boldsymbol{x}$, for $1 \leq i \leq A(\boldsymbol{x})$
$\lvert \boldsymbol{x} \rvert$	—	The length of $\boldsymbol{x}$
$\boldsymbol{x}_I$	—	The projection of $\boldsymbol{x}$ on the ordered indices of $I \subseteq [\lvert \boldsymbol{x} \rvert]$
σ^n	—	The sequence with n consecutive σ's, for a symbol $\sigma \in \mathbb{Z}_q$
$\mathcal{SCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau)$	—	The set of all shortest common supersequences of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau \in \mathbb{Z}_q^*$
$\mathsf{SCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau)$	—	The length of the shortest common supersequence (SCS) of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau$
$\mathcal{LCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau)$	—	The set of all longest common subsequences of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau$
$\mathsf{LCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau)$	—	The length of the longest common subsequence (LCS) of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau$
$\mathcal{C} \subseteq \mathbb{Z}_q^n$	—	A code
$\mathcal{DI}_{t_1,t_2}(\boldsymbol{x})$	—	The set of all words that can be obtained from $\boldsymbol{x}$ by t_1 deletions and t_2 insertions
$\boldsymbol{x}^{(\alpha)}$	—	An α-balanced sequence
A	—	$\arg\max_{1 \leq \alpha \leq n} \left\{ \left\lvert \mathcal{L}_1\left(\boldsymbol{x}^{(\alpha)}\right) \right\rvert \right\}$

$\zeta(\boldsymbol{x})$	—	The number of entries in $\boldsymbol{x}$ which are contained in exactly two alternating segments
$\boldsymbol{x}' \in \mathbb{Z}_q^{n-1}$	—	The difference vector of $\boldsymbol{x} \in \mathbb{Z}_q^n$
$\mathsf{Zeros}(\boldsymbol{x})$	—	The number of zeros in $\boldsymbol{x}$
$\chi(s)$	—	The number of maximal alternating segments of length s over all the sequences $\boldsymbol{x} \in \mathbb{Z}_q^n$
$\chi_1(s) \subseteq \chi(s)$	—	The number of alternating segments that do not overlap with the preceding segment and the succeeding segments
$\chi_2(s) \subseteq \chi(s)$	—	The number of alternating segments that overlap with the preceding segment and the succeeding segments
$\chi_3(s) \subseteq \chi(s)$	—	The number of alternating segments that overlap only with the succeeding segment
$\chi_4(s) \subseteq \chi(s)$	—	The number of alternating segments that overlap only with the preceding segment
$\mathcal{A} \subseteq \mathbb{Z}_q^n$	—	An anticode
$\mathcal{A}' \subseteq \mathbb{Z}_q^{n-1}$	—	The puncturing of an anticode $\mathcal{A} \subseteq \mathbb{Z}_q^n$ in the n-th coordinate

Abbreviations and Notations for Chapter 4

n	—	Sequence (word) length
Σ_q	—	The set of integers $\{0, 1, \ldots, q-1\}$
Σ_q^n	—	The set of all sequences of length n over the alphabet Σ_q
Σ_q^*	—	$\bigcup_{n=0}^{\infty} \Sigma_q^n$
$\lvert\boldsymbol{x}\rvert$	—	The length of $\boldsymbol{x}$
$[n]$	—	The set of integers $\{1, 2, \ldots, n\}$
$\boldsymbol{x}_{[i,j]}$	—	The substring of $\boldsymbol{x}$ from index i to index j, i.e., $x_i x_{i+1} \cdots x_j$
$\boldsymbol{x}_U$	—	The projection of $\boldsymbol{x} \in \Sigma_q^n$ on the ordered indices of $U \subseteq [n]$
$d_L(\boldsymbol{x}, \boldsymbol{y})$	—	The Levenshtein distance between two words $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^*$
$D_t(\boldsymbol{x})$	—	The t-deletion ball centered at $\boldsymbol{x}$
$I_t(\boldsymbol{x})$	—	The t-insertion ball centered at $\boldsymbol{x}$
$\rho(\boldsymbol{x})$	—	The number of runs in $\boldsymbol{x}$
$\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$	—	$I_{n-\lvert\boldsymbol{y}_1\rvert}(\boldsymbol{y}_1) \cap D_{\lvert\boldsymbol{y}_2\rvert-n}(\boldsymbol{y}_2)$, for $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \Sigma_q^*$ and $n \in \mathbb{N}$ such that $\lvert\boldsymbol{y}_1\rvert \leq n \leq \lvert\boldsymbol{y}_2\rvert$
$\lvert U\rvert$	—	The cardinality of the set U
$\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2)$	—	$\begin{cases} 0 & d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) > 2 \\ 1 & d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) = 2 \text{ and } \boldsymbol{y}_2 \xrightarrow{1} \boldsymbol{y}_1\,, \\ 2 & d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) = 2 \text{ and } \boldsymbol{y}_2 \xrightarrow{2} \boldsymbol{y}_1 \end{cases}$ for $\boldsymbol{y}_1 \in \Sigma_q^{n-1}, \boldsymbol{y}_2 \in \Sigma_q^{n+1}$,where $\boldsymbol{y}_2 \xrightarrow{i} \boldsymbol{y}_1$ denotes the case where $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{y}_2$ by deletion(s) from i run(s).
$\mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2)$	—	If $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{y}_2$ by deleting two consecutive symbols and shortening a maximal alternating segment of m symbols in $\boldsymbol{y}_2$ to a maximal alternating segment of $m-2$ symbols in $\boldsymbol{y}_1$ then $\mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) = m-2$, and $\mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 0$ otherwise
$\mathcal{CS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t)$	—	The set of all common subsequences of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t$
$\mathsf{LCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t)$	—	The length of the longest common subsequence (*LCS*) of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t$
$\mathcal{LCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t)$	—	The set of all longest common subsequences of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t$
$\mathcal{U}(\boldsymbol{y}_1, \boldsymbol{y}_2)$	—	$\{U \subseteq [\lvert\boldsymbol{y}_2\rvert] : (\boldsymbol{y}_2)_U = \boldsymbol{y}_1\}$
$\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2) \subseteq \mathcal{U}(\boldsymbol{y}_1, \boldsymbol{y}_2)$	—	The set of all right-most index sets of $\boldsymbol{y}_2$ in $\mathcal{U}(\boldsymbol{y}_1, \boldsymbol{y}_2)$
$\mathsf{LCS}(i, j)$	—	The length of the LCS of $\boldsymbol{x}_{[i]}$ and $\boldsymbol{y}_{[j]}$, for two words $\boldsymbol{x}$ and $\boldsymbol{y}$

Abbreviations and Notations for Chapter 5

Σ	—	A finite alphabet of size q
Σ_q^n	—	The set of all sequences of length n over the alphabet Σ_q
Σ^*	—	The set of all finite strings over Σ
$[n]$	—	The set of integers $\{0, 1, \ldots, n-1\}$
$\lvert\boldsymbol{x}\rvert$	—	The length of $\boldsymbol{x}$
$\mathrm{supp}(\boldsymbol{x})$	—	$\{i \in [n] : x_i \neq 0\}$
$\lVert\boldsymbol{x}\rVert$	—	$\lvert\mathrm{supp}(\boldsymbol{x})\rvert$
$\boldsymbol{x} \circ \boldsymbol{y}$	—	The concatenation of $\boldsymbol{x}$ and $\boldsymbol{y}$
$\{\!\{a, a, b, \ldots\}\!\}$	—	A multiset
$\mathcal{X}_{n,k}$	—	$\{S = \{\!\{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{k-1}\}\!\} : \forall i, \boldsymbol{x}_i \in \Sigma^n\}$
$L_{\min}$	—	The minimum length of a substring
$L_{\max}$	—	The maximum length of a substring
$\mathcal{T}_{L_{\min}}^{L_{\max}}(\boldsymbol{x})$	—	The $(L_{\min}, L_{\max})$-segmentation spectrum of a string $\boldsymbol{x}$
$\mathcal{T}_{L_{\min}}^{L_{\max}}(S)$	—	The $(L_{\min}, L_{\max})$-segmentation spectrum of a set of strings S
$\mathcal{T}_{\ell}(\boldsymbol{x})$	—	$\mathcal{T}_{\ell}^{\ell}(\boldsymbol{x})$
$\mathcal{T}_{\ell}(S)$	—	$\mathcal{T}_{\ell}^{\ell}(S)$
$\mathcal{C} \subseteq \mathcal{X}_{n,k}$	—	A code
$R(\mathcal{C})$	—	The rate of $\mathcal{C}$
$\mathrm{red}(\mathcal{C})$	—	The redundancy of $\mathcal{C}$
$\log$	—	The base-q logarithm
$\boldsymbol{x} \perp^s \boldsymbol{y}$	—	Denotes the case where $\boldsymbol{x}, \boldsymbol{y}$ have no common s-segment
$\boldsymbol{x}^{(i)}$	—	The s-segment of $\boldsymbol{x}$ at location i
$\mathcal{O}_{\boldsymbol{p}}$	—	$\left\{\boldsymbol{c} \in \Sigma^{n/m} : \boldsymbol{c} \perp^s \boldsymbol{p}\right\}$
$(\boldsymbol{c}_i)_{i \in [q^I]}$	—	For an integer I, $\boldsymbol{c}_i \in \Sigma^I$ are the codewords of a q-ary Gray code, in order
$\boldsymbol{c}_i'$	—	The concatenation of $\boldsymbol{c}_i$ with a single parity symbol
$\boldsymbol{c}_i''$	—	The result of inserting '1's into $\boldsymbol{c}_i'$ at every location divisible by $f(n)$
$\mathrm{Ind}(\boldsymbol{u})$	—	The index of $\boldsymbol{u}$
$\mathcal{BT}_{L_{\min}}^{L_{\max}}(\boldsymbol{x}; t)$	—	The t-error torn-paper ball centered at $\boldsymbol{x}$
$B_t(\boldsymbol{x})$	—	The radius-t Hamming ball centered at $\boldsymbol{x}$
$\mathcal{T}_{L_{\min}}^{+}(\boldsymbol{u})$	—	The multiset of non-overlapping $L_{\min}$-segments of $\boldsymbol{u}$, where the last segment is of length ℓ, $L_{\min} \leq \ell < 2L_{\min}$
$\mathcal{T}_{L_{\min}}^{+}(\mathcal{U})$	—	$\{\!\{\mathcal{T}_{L_{\min}}^{+}(\boldsymbol{u}) : \boldsymbol{u} \in \mathcal{U}\}\!\}$
$\mathrm{Ind}'(\boldsymbol{w})$	—	The decoding of the selected encoded index using Algorithm 4
$\mathcal{Z}(\mathcal{U})$	—	$\left\{(\mathrm{Ind}'(\boldsymbol{w}), \boldsymbol{w}) : \boldsymbol{w} \in \mathcal{T}_{L_{\min}}^{+}(\mathcal{U}) \text{ is valid}\right\}$
$\mathcal{Z}'(\mathcal{U}) \subseteq \mathcal{Z}(\mathcal{U})$	—	The subset of $\mathcal{Z}(\mathcal{U})$ that includes only the shortest, lexicographically-least, segments for each index
$\mathcal{DT}_{L_{\min}}^{L_{\max}}(\boldsymbol{x}; t)$	—	The t-deletion torn-paper ball centered at $\boldsymbol{x}$
$\mathcal{B}_{\mathrm{BE}}^{L}(\boldsymbol{x}; t)$	—	The t-burst L-erasures ball centered at $\boldsymbol{x}$

Abbreviations and Notations for Chapter 6

Σ_q	—	$\{0, 1, \ldots, q-1\}$, a finite alphabet of size q
Σ_q^n	—	The set of all sequences of length n over the alphabet Σ_q
$w(\boldsymbol{x})$	—	The Hamming weight of a sequence $\boldsymbol{x}$
$\boldsymbol{x} \circ \boldsymbol{y}$	—	The concatenation of $\boldsymbol{x}$ and $\boldsymbol{y}$
$\#_\sigma(\boldsymbol{x})$	—	The number of occurrences of the symbol $\sigma \in \Sigma_q$ in $\boldsymbol{x}$
$I_{\boldsymbol{x}} \subseteq [0, 1)$	—	The interval that represents the sequence $\boldsymbol{x}$
$f_p^{(\mathbf{ac})}$	—	The encoder of the arithmetic coding with parameter $p \in (0, 1)$
$g_p^{(\mathbf{ac})}$	—	The decoder of the arithmetic coding with parameter $p \in (0, 1)$
$\mathcal{C}(n, \varepsilon(n))$	—	$\left\{\boldsymbol{x} \in \Sigma_2^n \;\middle|\; w(\boldsymbol{x}) \in \left[\frac{n}{2} - \varepsilon(n), \frac{n}{2} + \varepsilon(n)\right]\right\}$
$F(n, \alpha)$	—	$\frac{\lvert\mathcal{C}(n, \alpha\sqrt{n})\rvert}{2^n}$
$\mathcal{C}_L(n, \alpha\sqrt{n})$	—	$\left\{\boldsymbol{x} \in \Sigma_2^n \;\middle|\; w(\boldsymbol{x}) \leq \frac{n}{2} + \alpha\sqrt{n}\right\}$
$\mathcal{C}_H(n, \alpha\sqrt{n})$	—	$\left\{\boldsymbol{x} \in \Sigma_2^n \;\middle|\; w(\boldsymbol{x}) \geq \frac{n}{2} - \alpha\sqrt{n}\right\}$
$d_{in}(\boldsymbol{v})$	—	The in-degree of a node $\boldsymbol{v}$
$\mathcal{C}_q^{(\mathbf{pb})}(n, \varepsilon(n))$	—	$\left\{\boldsymbol{x} \in \Sigma_q^n \;\middle|\; \sum_{i=0}^{\frac{q}{2}-1} \#_i(\boldsymbol{x}) \in \left[\frac{n}{2} - \varepsilon(n), \frac{n}{2} + \varepsilon(n)\right]\right\}$
$F^{(\mathbf{pb})}(n, \alpha)$	—	$\frac{\lvert\mathcal{C}_q^{(\mathbf{pb})}(n, \alpha\sqrt{n})\rvert}{q^n}$
$\mathcal{C}_{q,L}^{(\mathbf{pb})}(n, \alpha\sqrt{n})$	—	$\left\{\boldsymbol{x} \in \Sigma_q^n \;\middle|\; \sum_{i=0}^{\frac{q}{2}-1} \#_i(\boldsymbol{x}) \leq \frac{n}{2} + \alpha\sqrt{n}\right\}$
$\mathcal{C}_{q,H}^{(\mathbf{pb})}(n, \alpha\sqrt{n})$	—	$\left\{\boldsymbol{x} \in \Sigma_q^n \;\middle|\; \sum_{i=0}^{\frac{q}{2}-1} \#_i(\boldsymbol{x}) \geq \frac{n}{2} - \alpha\sqrt{n}\right\}$
$f_{q,p}^{(\mathbf{pb\text{-}ac})}$	—	The encoder of the modified arithmetic coding with parameter $p \in (0, 1)$
$g_{q,p}^{(\mathbf{pb\text{-}ac})}$	—	The decoder of the modified arithmetic coding with parameter $p \in (0, 1)$
$\mathcal{C}_4^{(\mathbf{sb})}(n, \varepsilon(n))$	—	$\left\{\boldsymbol{x} \in \Sigma_4^n \;\middle|\; \#_\sigma(\boldsymbol{x}) \in \left[\frac{n}{4} - \varepsilon(n), \frac{n}{4} + \varepsilon(n)\right], \forall \sigma \in \Sigma_4\right\}$
$\mathcal{C}_{0,i}^{(\mathbf{pb})}$	—	$\left\{\boldsymbol{x} \in \Sigma_4^n \;\middle|\; \#_0(\boldsymbol{x}) + \#_i(\boldsymbol{x}) \in \left[\frac{n}{2} - \frac{\alpha\sqrt{n}}{2}, \frac{n}{2} + \frac{\alpha\sqrt{n}}{2}\right]\right\}$
$f_{4,p,i}^{(\mathbf{pb\text{-}ac})}$	—	The encoder of the modified arithmetic coding with parameter $p \in (0, 1)$ that associates the first two intervals in each partition with 0 and i
$g_{4,p,i}^{(\mathbf{pb\text{-}ac})}$	—	The decoder of the modified arithmetic coding with parameter $p \in (0, 1)$ that associates the first two intervals in each partition with 0 and i
$\mathcal{C}_{L,0,i}^{(\mathbf{pb})}$	—	$\left\{\boldsymbol{x} \in \Sigma_4^n \;\middle|\; \#_0(\boldsymbol{x}) + \#_i(\boldsymbol{x}) \leq \frac{n}{2} + \frac{\alpha\sqrt{n}}{2}\right\}$
$\mathcal{C}_{H,0,i}^{(\mathbf{pb})}$	—	$\left\{\boldsymbol{x} \in \Sigma_4^n \;\middle|\; \#_0(\boldsymbol{x}) + \#_i(\boldsymbol{x}) \geq \frac{n}{2} - \frac{\alpha\sqrt{n}}{2}\right\}$

Abbreviations and Notations for Chapter 7

$\Sigma = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$	—	The DNA alphabet
Σ^n	—	The set of all sequences of length n over the alphabet Σ
Σ^*	—	The set of all sequences of any length over Σ
$[n]$	—	The set of integers $\{1, 2, \ldots, n\}$
$S = (s_1, \ldots, s_n) \in \Sigma^n$	—	A reference sequence
$S_{[i;\ell]}$	—	$(s_i, \ldots, s_{i+\ell-1})$, for $1 \leq i \leq n - \ell + 1$
$W_\ell(S)$	—	$\{S_{[i;\ell]} : 1 \leq i \leq n - \ell + 1\}$
$\Lambda \in \Sigma^*$	—	A label
$\Lambda(S)$	—	The λ-labeling of a sequence S
t, T	—	Number of labels
$\Lambda = \{\lambda_1, \ldots, \lambda_t\}$	—	A set of t labels
$\Lambda(S)$	—	$\lambda_1(S) \vee \lambda_2(S) \vee \cdots \vee \lambda_t(S)$
$\Lambda_1 \equiv_S \Lambda_2$	—	The labeling of S with Λ_1 is identical to the labeling of S with Λ_2
$M(S)$	—	$\max\{\ \lvert\mathcal{C}\rvert\ :\ \mathcal{C}$ is S-uniquely-decodable$\}$
$\mathcal{C}_S$	—	An S-uniquely-decodable code with maximum size, i.e. $\lvert\mathcal{C}_S\rvert = M(S)$
$\mathcal{V} = \{\lambda_1, \lambda_2, \ldots, \lambda_T\}$	—	Executable labels
$\mathcal{P}(\mathcal{V})$	—	the power set of $\mathcal{V}$
$M(S, \mathcal{V})$	—	$\max\{\lvert\mathcal{C}\rvert\ \lvert\mathcal{C}$ is S-uniquely-decodable, $\mathcal{C} \subseteq \mathcal{P}(\mathcal{V})\}$
$M(n, \mathcal{V})$	—	$\max_{S \in \Sigma^n}\{M(S, \mathcal{V})\}$
$S_{(n,\mathcal{V})} \in \Sigma^n$	—	A reference sequence such that $M(S_{(n,\mathcal{V})}, \mathcal{V}) = M(n, \mathcal{V})$
$\mathcal{C}_{(n,\mathcal{V})} \subseteq \mathcal{P}(\mathcal{V})$	—	An $S_{(n,\mathcal{V})}$-uniquely decodable code
σ	—	A symbol in Σ
$\pi(S)$	—	The minimal period of the sequence S
ℓ	—	Label length
$M_\ell(S)$	—	$M(S, \Sigma^\ell)$
$M_\ell(n)$	—	$M(n, \Sigma^\ell)$
$S_{(n,\ell)}$	—	$S_{(n,\Sigma^\ell)}$
$\mathcal{C}_{(n,\ell)}$	—	$\mathcal{C}_{(n,\Sigma^\ell)}$
$\mathcal{T}_\ell$	—	The set of all binary sequences in which any run of *ones* is of length at least ℓ
RLL	—	Run-Length Limited
$\mathcal{C}_{d,k}(n)$	—	The set of all sequences of length n that satisfy the (d,k)-RLL constraint
$\mathcal{T}_\ell(n)$	—	$\mathcal{T}_\ell \cap \{0,1\}^n$
X, Y	—	Binary sequences ($X, Y \in \mathcal{T}_\ell(n)$)
Λ_X	—	The encoding of X obtained by Construction 7.1
A	—	The set of binary strings of length n in which each run of *ones* is of length at least ℓ
A^0	—	$\{(x_1, \ldots, x_n) \in A :\ x_1 = 0\}$
A^1	—	$\{(x_1, \ldots, x_n) \in A :\ x_1 = 1\}$
$\lvert A\rvert$	—	The cardinality of the set A
A^0_r, A^0_r	—	The set of all sequences in A^0, A^1, respectively, that have exactly r runs
r_0, r_1	—	The number or runs of *zeros, ones*, respectively (in a binary sequence)
$\left(\!\binom{m}{b}\!\right)$	—	The number of b-element combinations of m objects, with repetitions
$\mathcal{C}_{(S)}$	—	$\{\Lambda \cap W_\ell(S) :\ \Lambda \in \mathcal{C}\}$
$\mathcal{N}_\ell$	—	Non-overlapping code of length ℓ over Σ
$\mathcal{N}_{(\ell,S)}$	—	$\mathcal{N}_\ell \cap W_\ell(S)$

Abbreviations and Notations for Chapter 8

DNN	—	Deep Neural Networks
ECC	—	Error-Correcting Codes
TP	—	Tensor-Product
$\Sigma = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$	—	The DNA alphabet
PCR	—	Polymerase Chain Reaction
RS	—	Reed Solomon
CPL	—	Conditional Probability Logic
BMA	—	Bitwise Majority Alignment
SOTA	—	State-of-the-art
α	—	The number of erroneous predictions with less than 4 errors in the decoder's input
β	—	The number of erroneous predictions with at least 4 errors in the decoder's input
γ	—	The number of missing predictions in the decoder's input
GPU	—	Graphics Processing Unit
NCI	—	Non-Coherent Integration
SNR	—	Signal to Noise Ratio
λ_i	—	Hyperparameters of the loss function
Θ_i	—	The model prediction probabilities
$y_n, y[n]$	—	The n-th symbol of the vector $\boldsymbol{y}$
L	—	The length of encoded sequences
M	—	The output of the DNN (a $4 \times L$ matrix)
$M_{i,j}$	—	The value of the i-th entry in the j-th column of M
$m(M)$	—	$\frac{1}{L}\sum_{j=1}^{L} \max\{M_{1,j}, M_{2,j}, M_{3,j}, M_{4,j}\}$
$c(M, \textit{cluster size})$	—	$m(M)^{2 \cdot \textit{cluster size}}$
$\mathbf{C}$	—	The input to the CPL algorithm (a cluster)
t	—	The number of reads in a cluster
$\boldsymbol{y}_i$	—	The reads in $\mathbf{C}$
$\mathbf{EV}(\boldsymbol{y}_1, \boldsymbol{y}_k)$	—	A vector of edit operations that describes how to obtain $\boldsymbol{y}_k$ from $\boldsymbol{y}_1$
$\mathbf{EV}_{\boldsymbol{y}_1}(\mathbf{C})$	—	The set of all vectors $\mathbf{EV}(\boldsymbol{y}_1, \boldsymbol{y}_k)$ (for $2 \leq k \leq t$)
$\lvert\boldsymbol{y}\rvert$	—	The length of the vector $\boldsymbol{y}$
$\mathbf{EV}_{CDS}(\boldsymbol{y}_1, \boldsymbol{y}_k), \mathbf{EV}_I(\boldsymbol{y}_1, \boldsymbol{y}_k)$	—	An auxiliary vectors used by the CPL algorithm
$p_i^{\boldsymbol{y}_{j'}, \mathbf{C}}(\sigma \lvert \sigma')$	—	An estimated conditional probability used by the CPL algorithm
$Post_{i,\sigma'}^{\boldsymbol{y}_{j'}, \mathbf{C}}$	—	A set of symbols that can be achieved on the $(i+1)$-th index of an error vector in $\mathbf{EV}_{\boldsymbol{y}_j}(\mathbf{C})$, given that the i-th symbol was σ'
$G_{\boldsymbol{y}_j, \mathbf{C}}$	—	An edit graph used by the CPL algorithm
b	—	The number of bits in the encoder's input
M	—	The number of encoded sequences in the decoder's output
u	—	The length of the encoded sequences excluding the index
E_{Ind}, D_{Ind}	—	Index encoder and decoder
E_{DRS}, D_{DRS}	—	Diagonal-column encoder and decoder
E_{Cons}, D_{Cons}	—	Encoder and Decoder of the constrained code
E_{TP}, D_{TP}	—	Encoder and Decoder of the TP code

BCH	—	Bose–Chaudhuri–Hocquenghem
r_1, r_2, r_3	—	The redundancy of the Diagonal-column, BCH, RS code, respectively
$\boldsymbol{x}$	—	The binary data (the input for the encoder)
X	—	The codeword matrix
b_ℓ, b_s	—	Parameters of the encoder (used for input processing)
$\mathcal{I}$	—	Matrix in which the i-th row contains the i-th index
$\mathcal{X}_b$	—	An auxiliary matrix for the encoder that contains the indices and the binary input
A', A'', B'	—	Sub-matrices of $\mathcal{X}_b$ which correspond to the binary input and are used by the encoder
C'	—	A Sub-matrix of $\mathcal{X}_b$ which corresponds to the E_{DRS}
A, B, C	—	The sub-matrices which correspond to the encoding of A', A'', B' and C' using E_{Cons}
$\mathbf{H}$	—	A parity-check matrix of a BCH code over the 4-ary alphabet (can correct 3 substitution errors)
$\mathcal{X}$	—	The input to E_{TP}
S	—	An auxiliary matrix used by E_{TP} and D_{TP}
$\hat{X}, \hat{S}, \hat{A}, \hat{B}, \hat{C}\hat{A}', \hat{B}', \hat{C}'$	—	Erroneous versions of X, S, A, B, C, A', B', and C' used by the encoder

Abbreviations and Notations for Chapter 9

CCP	—	Coupon Collector's Problem
$\Sigma = \{A, C, G, T\}$	—	The DNA alphabet
ℓ	—	Vector length
Σ^ℓ	—	The set of all sequences of length ℓ over the alphabet Σ
Σ^*	—	$\bigcup_{\ell=0}^{\infty} \Sigma^\ell$
$[n]$	—	The set of integers $\{1, 2, \ldots, n\}$
$\mathcal{C}$	—	A code
$\mathbf{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k) \in (\Sigma^\ell)^k$	—	The input to the encoder (information sequences)
$\mathbf{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) \in (\Sigma^\ell)^n$	—	The output from the encoder (encoded sequences)
$\mathbf{U} \circ \mathbf{V}$	—	The concatenation of $\mathbf{U}$ and $\mathbf{V}$
(n, k)	—	Code parameters
$[n, k]$	—	MDS code parameters
S	—	The DNA storage channel
M	—	Sample size
$\mathcal{Y}_M = \{\!\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_M\}\!\}$	—	The output of the DNA storage channel
NGS	—	Next-Generation Sequencing
$\boldsymbol{p} = (p_1, \ldots, p_n)$	—	The channel probability distribution, where p_i is the probability to sample a read of $\boldsymbol{x}_i$
t	—	The minimal number of reads needed for reconstructing a cluster
$\nu_t^{\boldsymbol{p}}(\mathcal{C})$	—	The random variable that governs the number of reads that should be sampled for successful decoding of $\mathbf{U}$
$\nu_t^{\boldsymbol{p}}(n, k)$	—	$\nu_t^{\boldsymbol{p}}(\mathcal{C})$ for the case $\mathcal{C}$ is an $[n, k]$ MDS code
$\boldsymbol{p}_u$	—	$(\frac{1}{n}, \ldots, \frac{1}{n})$
$\nu_t(\mathcal{C})$	—	$\nu_t^{\boldsymbol{p}_u}(\mathcal{C})$
$\nu_t(n, k)$	—	$\nu_t^{\boldsymbol{p}_u}(n, k)$
$\tau_i(\mathcal{C})$	—	The random variable that governs the number of samples to recover the i-th information strand assuming noiseless channel with uniform distribution
$T_{\max}^{\mathcal{C}}$	—	$\max_{1 \le i \le k} \mathbb{E}[\tau_i(\mathcal{C})]$
$T_{\max}$	—	$T_{\max}^{\mathcal{C}}$ in the case where no coding is uded
$T_{\text{avg}}^{\mathcal{C}}$	—	$\frac{1}{k} \sum_{i=1}^{k} \mathbb{E}[\tau_i(\mathcal{C})]$
T_{avg}	—	$T_{\text{avg}}^{\mathcal{C}}$ in the case where no coding is uded
H_n	—	The n-th harmonic number
γ	—	The Euler–Mascheroni constant
$e_t(x)$	—	$\sum_{i=0}^{t} \frac{x^i}{i!}$
$[u^q]Q(u)$	—	The coefficient of u^q in a polynomial $Q(u)$
$R = \frac{k}{n}$	—	Code rate
$\mathsf{supp}(\boldsymbol{p})$	—	The support of the vector $\boldsymbol{p}$
$r(n, k, t)$	—	$n \log\left(\frac{n}{n-k}\right) + nt \log\log n + 2n \log(t+1)$
$E_t^{(r)}$	—	The event where after r rounds, there exists a set S_t, of $n-k+1$ urns, each containing less than t balls
$z_i(n, r)$	—	The random variable that governs the number of balls in the i-th urn, after r draws
Y	—	The random variable that governs the number of urns with less than t balls

$r_f(n, k = Rn, t)$	—	$n \log\left(\frac{1}{1-R}\right) + ntf(n) + 2n(t+1)$
$r_L(n, k, c)$	—	$n \log\left(\frac{n}{n-k}\right) - nc$
$G_{\nu_1(n,k)}(x)$	—	The generating function of the geometric random variable $\nu_1(n, k)$
B_h	—	The h-th Bernoulli number
$X^{(r)}$	—	The random variable that governs the number of urns that are not filled with at least t balls after r rounds
$D(a\|\|p)$	—	The Kullback–Leibler divergence
$r_E(n, k = Rn, t)$	—	$n(t-1) - n\log 2 \log(1-R) + n(t-1)\sqrt{-\frac{2\log 2}{t-1}\log(1-R)}$
$\omega_\alpha(n, k)$	—	The random variable describing the required sample size to ensure successful decoding in case a single noiseless copy from k out of the n synthesized strands
$\widehat{\tau}_i(\mathcal{C})$	—	The random variable governs the required sample size to retrieve the i-th encoded strand
$\widehat{\tau}_J(\mathcal{C})$	—	$\max_{i\in J} \widehat{\tau}_i(\mathcal{C})$, for $J \subseteq [n]$
$\widehat{\mathcal{D}}(i)$	—	The set of all retrieval sets of $\boldsymbol{u}_i$
$\mathcal{D}(i)$	—	The set of all minimal retrieval sets of $\boldsymbol{u}_i$ (with respect to the inclusion relation)
$\mathcal{E}_{r-1}$	—	The random variable that represents the number of unique strands that were sampled in the first $r-1$ draws
$\mathcal{C}_{(2k,k)}$	—	The $(2k, k)$ code obtained by Construction 9.1
$\mathcal{C}^\gamma$	—	The γ-block code of $\mathcal{C}$
$E_{\mathcal{C}}$	—	The encoder of the code $\mathcal{C}$
$\mathcal{C}^{\mathrm{MDS}}_{n,k,p}$	—	An $[n(1-p)+k, k]$ systematic MDS code
$\mathcal{C}^{k}_{n,p}$	—	The (n, k) code obtained by Construction 9.2
$n_i(\boldsymbol{v})$,	—	The minimum read index h which allows retrieving the i-th information strand $\boldsymbol{u}_i$, given a sequence of reads $\boldsymbol{v}$
$t_i(v)$	—	The time to collect the i-th new sample (after collecting the previous one), given a sequence of reads $\boldsymbol{v}$
$t_j(\mathcal{C})$	—	The random variable that governs the time to collect the j-th new sample (after collecting the previous one)
$W_{k,n}$	—	The random variable that represents the number of samples needed to obtain k distinct coupons where each draw is taken from a pool of n total coupons
$W_{k,n}(x)$	—	The generating function for $W_{k,n}$
$V_{r,n}$	—	The random variable that represents the number of distinct coupons in the first r draws, where each coupon is taken from a pool of n total coupons
$D_{k,i,n}$	—	The random variable that represents the required number of draws to obtain k distinct coupons or to retrieve coupon i (whichever occurs first), where each draw is taken from a pool of n total coupons
$D_{k,i,n}(x)$	—	The generating function for $D_{k,i,n}$
$D^{(j)}_{k,i,n}$	—	For $0 \le j \le k-1$, the random variable that represents the number of samples needed to obtain j distinct coupons (each not equal to the i-th coupon), followed by the i-th coupon
$D^{(k)}_{k,i,n}$,	—	The random variable that represents the number of samples needed to obtain k distinct coupons (each not equal to the i-th coupon)
ψ	—	The digamma function

A Note to the Reader

This dissertation is based on the following publications.

Journal Papers

[J1] **D. Bar-Lev**, T. Etzion, and E. Yaakobi. "On the Size of Balls and Anticodes of Small Diameter under the Fixed-Length Levenshtein Metric," *IEEE Transactions on Information Theory*, vol. 69, no. 4, pp. 2324-2340, April 2023.

[J2] **D. Bar-Lev**, S. Marcovich, E. Yaakobi, and Y. Yehezkeally. "Adversarial Torn-Paper Codes," *IEEE Transactions on Information Theory*, vol. 69, no. 10, pp. 6414-6427, October 2023.

[J3] **D. Bar-Lev**, I. Orr, O. Sabary, T. Etzion, and E. Yaakobi "Deep DNA Storage: Scalable and Robust DNA Storage via Coding Theory and Deep Learning," submitted to *Nature Machine Intelligence*.

[J4] **D. Bar-Lev**, O. Sabary, R. Gabrys, and E. Yaakobi. "Cover Your Bases: How to Minimize the Sequencing Coverage in DNA Storage Systems," submitted to *IEEE Transactions on Information Theory*.

Peer-Reviewed Conference Proceeding

[C1] **D. Bar-Lev**, T. Etzion, and E. Yaakobi. "On Levenshtein Balls with Radius One," *IEEE International Symposium on Information Theory (ISIT)*, Melbourne, Australia, 12-20 July, 2021 (virtual). (Contained in [J1]).

[C2] **D. Bar-Lev**, O. Sabary, Y. Gershon and E. Yaakobi. "The Intersection of Insertion and Deletion Balls," *IEEE Information Theory Workshop (ITW)*, Kanazawa, Japan, 17-21 October, 2021 (virtual).

[C3] **D. Bar-Lev**, S. Marcovich, E. Yaakobi, Y. Yehezkeally. "Adversarial Torn-Paper Codes," *IEEE International Symposium on Information Theory (ISIT)*, Espoo, Finland, June 26-July 1, 2022. (Contained in [J2]).

[C4] **D. Bar-Lev**, O. Sabary, R. Gabrys, and E. Yaakobi. "Cover Your Bases: How to Minimize the Sequencing Coverage in DNA Storage Systems," *IEEE International Symposium on Information Theory (ISIT)*, Taipei, Taiwan, 25-30, June, 2023. (Contained in [J4]).

[C5] **D. Bar-Lev**, A. Kobovich, O. Leitersdorf, and E. Yaakobi. "Optimal Almost-Balanced Sequences," *IEEE International Symposium on Information Theory (ISIT)*, Athens, Greece, 7-12 July, 2024.

[C6] **D. Bar-Lev**, T. Etzion, E. Yaakobi, and Z. Yakhini. "Representing Information on DNA using Patterns Induced by Enzymatic Labeling," *IEEE International Symposium on Information Theory (ISIT)*, Athens, Greece, 7-12 July, 2024.

All research works presented here were conducted in collaboration with my supervisor, Prof. Eitan Yaakobi. Additionally, the works [J1], [J3], [C1], and [C6] were also done in collaboration with my supervisor, Prof. Tuvi Etzion. Prof. Etzion and Prof. Yaakobi introduced me to the field of Coding Theory and related topics, with Prof. Yaakobi specifically guiding me into the field of coding for DNA storage. They provided invaluable guidance throughout the research process, including referring me to relevant literature, helping define research problems, and suggesting methods and approaches for tackling these problems. Their insights were integral to the development of the results presented in this thesis, and they provided ongoing support throughout the research and writing process.

This dissertation consists of three main parts, each contributing to the DNA storage field. The first part of this dissertation focuses on the theoretical analysis of combinatorial structures related to *Insertions and Deletions Errors*, two common types of errors in DNA storage systems. Part I contains the published journal paper [J1] and its shortened conference version [C1], as well as the conference paper [C2]. The second part of the dissertation addresses *Coding for DNA Storage*, considering the unique characteristics of DNA storage systems, analyzing relevant channels, and presenting corresponding codes. Part II includes the published journal paper [J2] and its shortened conference version [C3], as well as the accepted conference paper [C5]. Lastly, the third part focuses on the challenges to overcome in order to advance *Towards Practical DNA Storage Systems*. Part III includes the results of the submitted journal papers [J3] and [J4] (along with its already published shortened conference version [C4]), as well as the results of the accepted conference paper [C6].

Contribution of the authors: All of the papers relate to this work's main research topic, which is investigating theoretical and practical problems relevant to the advancement of DNA-based storage systems.

The results of [J1] and its shortened conference version [C1] were derived by me in collaboration with my supervisors, Prof. Eitan Yaakobi and Prof. Tuvi Etzion.

The paper [C2] was done with Omer Sabary and Yotam Gershon under the supervision of Prof. Yaakobi. Most of the results in this paper were developed by me, while Mr. Sabary and Mr. Gershon suggested valuable ideas that assisted me with proving the results. The writing of the paper was done by Mr. Sabary and me, under the supervision of Prof. Yaakobi, and the implementation of the algorithm given in the paper was done by Mr. Sabary.

The work in [J2] and its shortened conference version [C3] was done with Dr. Sagi Marcovich and Dr. Yonathan Yehezkeally under the supervision of Prof. Yaakobi. Dr.

Marcovich, Dr. Yehezkeally, and I contributed the results equally. In this work, the significant contributions appear in Section 3 and Section 4. The first part of Section 3 was mainly done by Dr. Yehezkeally, and the second part, which presents the basic construction used throughout the paper, was written in equal partnership by Dr. Marcovich, Dr. Yehezkeally, and me. Section 4 is composed of two subsections, each considering a different error model. The results in the first subsection were developed and written by me, and the results in the second subsection were developed and written by Dr. Marcovich, under the supervision of Prof. Yaakobi.

The paper [J3] was done with Dr. Itai Orr and Omer Sabary under the supervision of Prof. Etzion and Prof. Yaakobi. Dr. Orr, Mr. Sabary, and I contributed the results equally. The idea for the multidisciplinary project presented in the paper was suggested by me and was developed mainly by me and Dr. Orr. The high-level integration between all the components in the work, the coding scheme, and the simulator was done by me. The safety margin mechanism was developed by Mr. Sabary and me together. The deep neural network architecture and its development and training were done by Dr. Orr. The wet experiments and their analysis, the CPL algorithm, and the implementation of most of the coding schemes were done by Mr. Sabary.

Paper [J4] and its shortened conference version [C4] were done in collaboration with Omer Sabary and under the supervision of Dr. Ryan Gabrys and Prof. Yaakobi. The results were contributed equally by Mr. Sabary, and me. The new problems defined in this project were defined by Mr. Sabary and me with the guidance and assistance of Dr. Gabrys and Prof. Yaakobi. The main results of this work appear in Section 4, Section 5, and Section 6. The work in Section 4 and Section 5 was done by Mr. Sabary and me together, where my main contributions are the results in Theorem 1 and Theorem 6 with their corresponding claims and lemmas; I also assisted with the results of Theorem 4 and Theorem 5, which were mainly developed by Mr. Sabary and Dr. Ryan Gabrys. The work in Section 6 was mostly done by me with the assistance and guidance of Dr. Gabrys and Prof. Yaakobi.

The work in [C5] was done together with Adir Kobovich and Orian Leitersdorf and under the supervision of Prof. Yaakobi. The results, constructions, and most of the writing were done by me with the guidance of Prof. Yaakobi. Mr. Kobovich and Mr. Leitersdorf suggested helpful discussion that allowed the development of the approach that was used in this work and is the topic of another paper we co-authored.

The paper [C6] was done with Prof. Etzion, Prof. Yaakobi, and Prof. Zohar Yakhini. The new model that is presented in the paper was suggested by me and was defined and formalized by me with the assistance of Prof. Yakhini. The results were obtained by me with the assistance and guidance of Prof. Etzion, Prof. Yaakobi, and Prof. Yakhini.

Following the instructions of the Technion's Graduate School, the structure of this dissertation is as follows. The first chapter, Introduction, presents a comprehensive and up-to-date overview of all areas of research. Afterwards, the Research Methods chapter describes the techniques developed and used in all of the papers. Then, we present

the conference and journal papers organized in the three parts described above, where the unpublished works appear last and each paper contains its own list of references.

The last chapter, Discussion, includes a brief summary of the results while taking into account the coherence and integration of the entire work. The separate bibliography which is given in p. 271 lists the references which are given in the Introduction, the Research Methods, and the Discussion chapters.

Chapter 1

Introduction

The amount of digital data being generated and stored is increasing rapidly. According to a study by the International Data Corporation (IDC), the global data sphere is expected to grow from 33 Zettabytes (ZB) in 2018 to 175 ZB by 2025, an increase of more than five times [18]. The demand for storage capacity already exceeds the supply, and the gap continues to grow exponentially. This growth is driven by various factors, including the increasing use of smartphones, the growth of the Internet, and the increasing adoption of technologies such as the Internet of Things (IoT) and artificial intelligence (AI). This explosion of data creates the need for more storage capacity and more effective ways to manage and analyze data. Hence, traditional storage methods, such as hard drives and tapes, are becoming less suitable for long-term data storage. This is where new technologies like DNA storage come in as a potential solution, offering high data density, durability, and security.

The idea of using DNA molecules for ultra-dense storage was first suggested in the 1960s by Richard Feynman in his famous talk, "There is plenty of room at the bottom" [26]. DNA possesses unique properties that make it well-suited for non-volatile recording media; among them are its high capacity and density and its robust stability. Additionally, recent technological advancements for writing and reading DNA, such as synthesizing artificial DNA and sequencing DNA, have greatly improved their efficiency and accuracy. As a result, it is anticipated that DNA-based storage systems will become a reality in the near future [21, 48].

In general, DNA consists of four structural chemical building blocks called *nucleotides* - adenine (A), cytosine (C), guanine (G), and thymidine (T), arranged in an ordered sequence to form a single DNA *strand* or *oligonucleotide*. With modern DNA synthesizers, DNA strands can be chemically synthesized by concatenating the four DNA nucleotides, allowing digital data to be stored in the strands. The length of the DNA strands is typically limited to 250-300 nucleotides to maintain an acceptable error rate, and each strand has multiple synthesized copies, where each of them is a noisy version of one of the designed strands. The synthesized strands are all stored together unordered in a storage container. Later, to read back the data, common DNA sequencers that use DNA polymerase enzymes or nanopore-based sequencing can be

used. To allow the error-free retrieval of the stored information, additional algorithmic and computational steps should be utilized. These include the encoding and decoding of the data using error-correcting codes (ECC) and constrained codes. Additionally, it is important to design retrieval algorithms that process the unordered set of reads from the sequencing machine as part of the decoding process. A schematic description of a DNA storage system can be found in Figure 1.1.

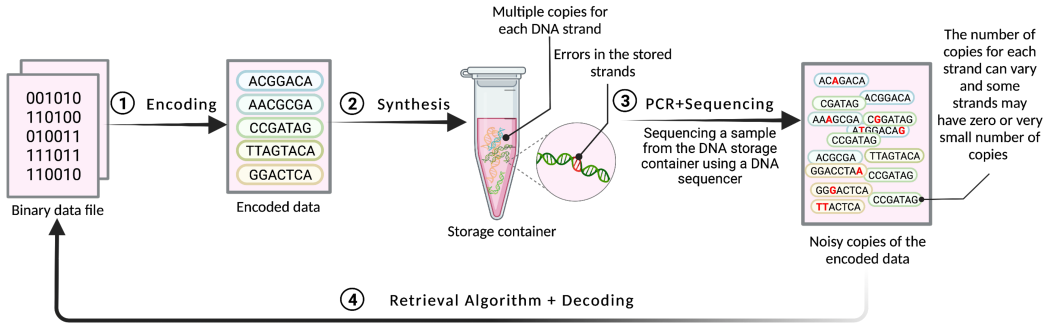


Figure 1.1: A schematic presentation of a DNA storage system. This figure was created with BioRender.com.

DNA storage has several unique attributes that distinguish it from its digital counterparts. The synthesis and sequencing processes introduce a unique error behavior dominated by deletions, substitutions, and insertions of symbols. Additionally, the strands are not ordered in the storage container, which means that the order in which they were stored is unknown. Every strand has multiple copies, which provides inherent redundancy that can be used during decoding. To use this inherent redundancy in the decoding process, the noisy copies should first be grouped by their original strands through a *clustering* step. The clustering is usually done by encoding *indices* on each of the strands as part of the coding scheme and then utilizing them in the clustering step. *Reconstruction* algorithms are then used to estimate the designed strand for each cluster, utilizing the inherent redundancy to correct errors. Finally, any remaining errors are corrected using the pre-defined ECC.

1.1 Deletion and Insertion Errors

The rapid development of DNA storage has brought the deletion and insertion channels to the front line of research. Part I highlights various aspects related to such errors through a couple of related works.

The size of a ball is one of the most fundamental parameters in any distance metric. A better understanding of the error balls corresponding to insertion and deletion types of errors is the first step in constructing codes that correct multiple deletions and

insertions.

For an integer $q \geq 2$, let Σ_q denote the q-ary alphabet $\{0, 1, \ldots, q-1\}$ and let Σ_q^n be the set of all sequences of length n over the alphabet Σ_q. For an integer t, $0 \leq t \leq n$, a sequence $\boldsymbol{y} \in \Sigma_q^{n-t}$ is a *t-subsequence* of $\boldsymbol{x} \in \Sigma_q^n$ if $\boldsymbol{y}$ can be obtained from $\boldsymbol{x}$ by deleting t symbols from $\boldsymbol{x}$. We say that $\boldsymbol{y}$ is a *subsequence* of $\boldsymbol{x}$ if $\boldsymbol{y}$ is a t-subsequence of $\boldsymbol{x}$ for some t. Similarly, a sequence $\boldsymbol{y} \in \Sigma_q^{n+t}$ is a *t-supersequence* of $\boldsymbol{x} \in \Sigma_q^n$ if $\boldsymbol{x}$ is a t-subsequence of $\boldsymbol{y}$ and $\boldsymbol{y}$ is a *supersequence* of $\boldsymbol{x}$ if $\boldsymbol{y}$ is a t-supersequence of $\boldsymbol{x}$ for some t.

Definition 1.1. *The radius-t deletion ball centered at $\boldsymbol{x} \in \Sigma_q^n$, $\mathcal{D}_t(\boldsymbol{x}) \subseteq \Sigma_q^{n-t}$, is the set of all t-subsequences of $\boldsymbol{x}$.*

Definition 1.2. *The radius-t insertion ball centered at $\boldsymbol{x} \in \Sigma_q^n$, $\mathcal{I}_t(\boldsymbol{x}) \subseteq \Sigma_q^{n+t}$, is the set of all t-supersequences of $\boldsymbol{x}$.*

Let $\boldsymbol{x} \in \Sigma_q^n$ be a sequence. The size of the insertion ball $|\mathcal{I}_t(\boldsymbol{x})|$ does not depend on $\boldsymbol{x}$ for any $0 \leq t \leq n$. To be exact, it was shown by Levenshtein [41] that

$$|\mathcal{I}_t(\boldsymbol{x})| = \sum_{i=0}^{t} \binom{n+t}{i} (q-1)^i. \tag{1.1}$$

While the size of the insertion ball was found by Levenshtein [41] more than 50 years ago, much less is known about the size of deletion balls, and even less is known about balls that correspond to the combination of these two types of errors. Calculating the exact size of the deletion ball is one of the more intriguing problems when studying codes for deletions. Deletion balls, unlike substitutions and insertions balls, are not regular. That is, the size of the deletion ball, $|\mathcal{D}_t(\boldsymbol{x})|$, depends on the choice of the sequence $\boldsymbol{x}$.

It was shown in [32] that the alternating sequences[1] have the largest deletion ball, denoted by $D_q(n,t)$, which is given by

$$D_q(n,t) = \sum_{i=0}^{t} \binom{n-t}{i} D_{q-1}(t, t-i)$$

where n is the sequence length, q is the alphabet size and t is the number of deletions. In particular, $D_2(n,t) = \sum_{i=0}^{t} \binom{n-t}{i}$ and $D_3(n,t) = \sum_{i=0}^{t} \binom{n-t}{i} \sum_{j=0}^{t-i} \binom{i}{j}$. The value $D_2(n,t)$ satisfies also the following recursion

$$D_2(n,t) = D_2(n-1,t) + D_2(n-2,t-1).$$

Definition 1.3. *A run is a maximal subsequence composed of consecutive identical symbols. For a sequence $\boldsymbol{x} \in \Sigma_q^n$, the number of runs in $\boldsymbol{x}$ is denoted by $\rho(\boldsymbol{x})$.*

[1] In the nonbinary case, an alternating sequence refers to a sequence that is composed of a periodical repetition of a permutation of all the alphabet symbols.

There are upper and lower bounds on the size of the deletion ball, which depend on the number of runs in the sequence [32,41,44].

Let $\Sigma_q^* \triangleq \bigcup_{n=0}^{\infty} \Sigma_q^n$ denote the set of all the sequences of any length. The *Levenshtein distance* between two words $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^*$, denoted by $d_L(\boldsymbol{x}, \boldsymbol{y})$, is the minimum number of insertions and deletions required to transform $\boldsymbol{x}$ into $\boldsymbol{y}$. Similarly, for two sequences $\boldsymbol{x}, \boldsymbol{y} \in \Sigma^*$, $d_E(\boldsymbol{x}, \boldsymbol{y})$ denotes the *edit* distance between $\boldsymbol{x}$ and $\boldsymbol{y}$, which is the minimum number of insertions, deletions and substitutions required to transform $\boldsymbol{x}$ into $\boldsymbol{y}$.

Definition 1.4. *Let $0 \leq t \leq n$ be integers. For a sequence $\boldsymbol{x} \in \Sigma_q^n$, the radius-$t$ Levenshtein ball centered at $\boldsymbol{x} \in \Sigma_q^n$, $\widehat{\mathcal{L}}_t(\boldsymbol{x})$, is defined by*

$$\widehat{\mathcal{L}}_t(\boldsymbol{x}) \triangleq \{\boldsymbol{y} \in \Sigma_q^* \ : \ d_L(\boldsymbol{x}, \boldsymbol{y}) \leq t\}.$$

In case $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^n$, for some integer n, the *FLL distance* between $\boldsymbol{x}$ and $\boldsymbol{y}$, $d_\ell(\boldsymbol{x}, \boldsymbol{y})$, is the smallest t for which there exists a t-subsequence $\boldsymbol{z} \in \Sigma_q^{n-t}$ of both $\boldsymbol{x}$ and $\boldsymbol{y}$, i.e.

$$d_\ell(\boldsymbol{x}, \boldsymbol{y}) = \min\{t' : \mathcal{D}_{t'}(\boldsymbol{x}) \cap \mathcal{D}_{t'}(\boldsymbol{y}) \neq \varnothing\}. \tag{1.2}$$

In other words, t is the smallest integer for which there exists $\boldsymbol{z} \in \Sigma_q^{n-t}$ such that $\boldsymbol{z} \in \mathcal{D}_t(\boldsymbol{x})$ and $\boldsymbol{y} \in \mathcal{I}_t(\boldsymbol{z})$. Note that if $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^n$ and $\boldsymbol{x}$ is obtained from $\boldsymbol{y}$ by t_1 deletions and t_2 insertions, then $t_1 = t_2$ which implies the following observation.

Observation 1.5. *For any integer $n \geq 0$ and sequences $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^n$, it holds that*

$$d_L(\boldsymbol{x}, \boldsymbol{y}) = 2d_\ell(\boldsymbol{x}, \boldsymbol{y}).$$

Definition 1.6. *Let $0 \leq t \leq n$ be integers. For a sequence $\boldsymbol{x} \in \Sigma_q^n$, the radius-$t$ FLL-ball centered at $\boldsymbol{x} \in \Sigma_q^n$, $\mathcal{L}_t(\boldsymbol{x}) \subseteq \Sigma_q^n$, is defined by*

$$\mathcal{L}_t(\boldsymbol{x}) \triangleq \{\boldsymbol{y} \in \Sigma_q^n \ : \ d_\ell(\boldsymbol{x}, \boldsymbol{y}) \leq t\}.$$

We say that a subsequence $\boldsymbol{x}_{[i,j]} \triangleq x_i x_{i+1} \cdots x_j$ is an *alternating segment* if $\boldsymbol{x}_{[i,j]}$ is a sequence of alternating distinct symbols $\sigma, \sigma' \in \mathbb{Z}_q$. Note that $\boldsymbol{x}_{[i,j]}$ is a *maximal alternating segment* if $\boldsymbol{x}_{[i,j]}$ is an alternating segment and $\boldsymbol{x}_{[i-1,j]}, \boldsymbol{x}_{[i,j+1]}$ are not. The number of maximal alternating segments of a sequence $\boldsymbol{x}$ will be denoted by $A(\boldsymbol{x})$.

Example 1.7. *If $\boldsymbol{x} = 0000000$ then $A(\boldsymbol{x}) = 7$ since $\boldsymbol{x}$ has seven maximal alternating segments, each of length one, and for $\boldsymbol{x} = 1120212$ we have that $A(\boldsymbol{x}) = 4$ and the maximal alternating segments are $1,\ 12,\ 202,\ 212$.*

The following formula to compute $|\mathcal{L}_1(\boldsymbol{x})|$ as a function of $\rho(\boldsymbol{x})$ and $A(\boldsymbol{x})$ was given in [63]

$$|\mathcal{L}_1(\boldsymbol{x})| = \rho(\boldsymbol{x}) \cdot (n(q-1)-1) + 2 - \sum_{i=1}^{A(\boldsymbol{x})} \frac{(s_i - 1)(s_i - 2)}{2}, \tag{1.3}$$

where s_i for $1 \leq i \leq A(\boldsymbol{x})$ denotes the length of the i-th maximal alternating segment of $\boldsymbol{x}$.

Part I focuses on these balls, where Chapter 3 presents a comprehensive discussion and exact computation on the balls with radius one and the anticodes with diameter one in the FLL metric. The chapter investigates fundamental parameters, such as minimum, maximum, and average ball sizes with radius one under the FLL metric over $\mathbb{Z}_q$, along with the analysis of maximal binary anticodes of diameter one. Additionally, Chapter 4 addresses the unexplored intersection of insertion and deletion balls, determining the maximum intersection size of any two insertion and deletion balls in the binary case. In the special scenario of one-insertion and one-deletion balls, we determine the intersection size for all pairs of sequences and derive the largest and average values of this intersection size. An efficient algorithm for computing the intersection of any t_1-insertion ball and t_2-deletion ball is also presented.

1.2 Coding for DNA Storage

DNA-based storage systems introduce new challenges that are unique to such systems and efficient coding techniques are paramount for ensuring reliable data retrieval and utilization. These challenges include DNA fragmentation, where information-carrying DNA strands may disintegrate during storage and retrieval processes, as well as the need for enforcing specific constraints such as almost-balanced GC content.

To address the challenge of DNA fragmentation, the torn-paper channel, [59, 66], also known as the *chop-and-shuffle channel* [55], is considered. The torn-paper channel involves segmenting a long information string into non-overlapping substrings with known distributions and then shuffling them. Previous studies have contributed insights into the torn-paper channel, focusing on probabilistic settings. The geometric distribution was first studied in [66], and later in [55]. Subsequently, [59] considered almost arbitrary distributions while, additionally, extending the problem by introducing incomplete coverage, i.e., assuming some of the substrings are deleted with some probability.

Our focus in Chapter 5 is on studying the adversarial torn-paper channel. This model extends the previously researched probabilistic setting to the worst-case scenario. Namely, it is assumed that an information string is adversarially segmented into non-overlapping substrings, where the length of each substring is between $L_{\min}$ and $L_{\max}$, for some given $L_{\min}$ and $L_{\max}$. Under this setup, code constructions for any parameters of the channel for which a non-vanishing asymptotic rate is possible are developed, and it is shown that our constructions achieve an asymptotically optimal rate while allowing for efficient encoding and decoding. Finally, the results are extended to related settings including multi-strand storage, presence of substitution errors, or incomplete coverage.

Continuing the exploration of coding strategies for DNA storage, in Chapter 6 we tend our attention toward constrained coding, which serves as a fundamental tool in

mitigating errors in any storage system. Under this topic, our focus is the almost-balanced sequences problem. The almost-balanced sequences problem generalized the CG-content constraint that is needed for DNA storage. During the storage phase in DNA strands, media degradation, and in particular breaks, can arise in DNA due to factors that include radiation, humidity, and high temperatures. One approach to dealing with media degradation is to generate strands of DNA that have approximately balanced GC-content (usually between 45% to 55% of GC ratio), and this approach has been leveraged in several existing works such as [24, 81, 83]. The construction of efficient balanced codes has been extensively studied; see e.g. [33, 37, 70, 71, 76], and extensions to non-binary balanced codes have been considered in [49, 50, 68, 77]. Codes that combine the balanced property with certain other constraints, such as run-length limitations, have also been addressed, for example, in [34].

Nevertheless, the problem of almost balanced sequences with Hamming weight between $0.5n \pm \varepsilon(n)$ has received much less attention. Under this framework, the goal is to find the optimal number of redundant bits as a function of $\varepsilon(n)$, where $\varepsilon(n)$ can be a function of n, e.g. linear in n, $\log n$, or a constant. No less important is the design of such algorithms. While Knuth's algorithm [37] is an efficient scheme to strictly balance an arbitrary sequence with $\log n + o(\log n)$ redundancy bits, designing an efficient encoder and decoder with less redundancy or even only a single bit is a non-trivial task. The best-known approach for minimizing $\varepsilon(n)$ is enumerative coding [16], which can achieve the lower bound of $\varepsilon(n) = \Omega(\sqrt{n})$, however optimizing it for efficient memory and time usage requires sophisticated and cumbersome techniques (see e.g., [61]). In contrast, simpler solutions with linear time complexity result in $\varepsilon(n)$ that is linear with n [57]. In Chapter 6, we present an explicit encoder that uses a single redundancy bit to balance binary sequences for $\varepsilon(n) = \Theta(\sqrt{n})$. On average our algorithm requires $\mathcal{O}(n)$ operations of rational numbers' multiplications.

1.3 Towards Practical DNA Storage Systems

The Zettabyte era, which we already experience in current days, is characterized by the explosive growth of digital data and requires advanced storage solutions capable of accommodating and processing such vast amounts of data. DNA emerges as a viable solution to accommodate that data growth, however, there are still several computational challenges that have to be addressed to make it possible.

Recall that the current synthesis methods typically limit the generated strands to a length of up to 300 bases [39]. Therefore, storing digital data in DNA requires splitting the information into small chunks, which are stored on multiple DNA molecules. Based on previous experiments and publications [7, 58], storing, for example, 1TB of information requires designing roughly 30 billion unique DNA strands. In addition to the high costs and low throughput of current synthesis and sequencing technologies, this amount of strands poses significant challenges in data retrieval, as it requires reading an enormous number of noisy reads, which is a linear multiplication of the number

of designed strands. Common approaches involve computationally expensive clustering and reconstruction methods, while other techniques that consider only noise-free reads also face challenges due to the extremely large required number of reads.

The special characteristics of DNA storage systems, such as the length limitation of synthesized strands, the need to store each information unit on multiple unordered strands, and the unique error characteristics, make the encoding and decoding tasks much more challenging compared to traditional data storage. Designing coding schemes that assist the data retrieval process and enhance the decoding performance are open problems that require further investigation. Another important factor that should be considered when designing a scalable DNA storage system is how to utilize the inherent redundancy of DNA synthesis using algorithmic methods and/or biological techniques, see e.g. [5], to improve both the information rate and the retrieval efficiency. It should be noted that since the inherent redundancy is unique to DNA storage systems, as of today, most practical solutions do not utilize this property.

Part III presents a series of contributions that address the computational and financial challenges in making DNA storage a practical solution for the Zettabyte era. Chapter 7 introduces a theoretical framework for an alternative representation of information using DNA molecules. Most existing DNA storage approaches rely on the 4-ary alphabet of DNA nucleotides, which requires DNA synthesis, a costly and time-consuming process. To overcome this limitation, alternative methods that can potentially eliminate the need for synthesis should be considered. Hence, inspired by DNA punch cards [69] and DNA composite [3,14,54], in Chapter 7, we present a theoretical framework for modeling DNA labeling, specifically tailored for data storage purposes. This approach involves leveraging patterns induced by a set of designed labels to represent information encoded into DNA molecules. The study explores various aspects of this labeling channel and provides upper bounds on the maximal size of the corresponding codes. An efficient encoder-decoder pair that is optimized for maximum code size under specific conditions is also presented.

In Chapter 8, DNAformer is introduced - a groundbreaking fusion of deep learning and coding theory techniques that significantly enhances error correction capabilities for DNA-based data retrieval. By harnessing the power of a Deep Neural Network trained on simulated data, DNAformer mitigates errors during synthesis, PCR, sequencing, and clustering, resulting in a 3200x increase in speed and 40% improvement in accuracy compared to current approaches. Lastly, Chapter 9 initiates the study of the DNA coverage depth problem, aiming to reduce the required number of sequencing reads and thereby lower the costs and latency of DNA storage. Through comprehensive analysis of the interplay between error correction codes, retrieval algorithms, and coverage depth, the work establishes theoretical bounds and explores optimal code-algorithm pairings, including for random-access setups.

Chapter 2

Research Methods

This chapter provides an overview of several tools and techniques which were used in this dissertation. These tools stem from the areas of combinatorics, constrained systems, coding theory, discrete mathematics, algorithms, machine learning, and graph theory. The following subsections specify which methods are used in each chapter.

Research Methods Invoked in Chapter 3

The methods used in Chapter 3 mainly stem from the fields of combinatorics, calculus and arithmetics. To analyze the size of the maximum FLL 1-balls we utilized the formula for the size of a 1-FLL ball that was given in [63] (see (1.3)). While in the non-binary case, the expression in (1.3) can be maximized using simple arithmetic steps, the analysis of the binary case required a more sophisticated approach. For this case, we first defined the concept of α-balanced sequences for an integer α and proved that among all the binary sequences of length n with exactly α alternating segments, the α-balanced sequences have the largest 1-FLL ball. Then, we found the optimal value of α using calculus and arithmetics. The average size of a 1-FLL ball was obtained by applying the linearity of the expectation on (1.3) and then using combinatorics and counting techniques to evaluate each term separately.

For the study of binary anticodes with diameter one under the FLL metric, we used a result by Levenshtein [43] regarding the maximum size of an intersection between two deletion balls with radius one or two insertion balls with radius one. By exploiting this result, we were able to prove several fundamental properties of anticodes in the FLL metric, which were then used in a thorough analysis of the maximum and minimum sizes of a maximal anticode with diameter one under this metric.

Research Methods Invoked in Chapter 4

The theoretical results on the size of the intersection of insertion and deletion balls in Chapter 4 are based on careful evaluation of all the possible cases of pairs of words,

considering their structure and distance from each other. For the size of the maximal intersection of binary insertion and deletion balls, we also exploit an upper bound on the size of a binary deletion ball that was given in [31, 41]. Our algorithm for computing the intersection of insertion and deletion balls is inspired by the dynamic programming implementation of the LCS problem [35]. To improve running time, we also used simple set theory arguments to work with an equivalent set of sequences, which is much smaller.

Research Methods Invoked in Chapter 5

In Chapter 5, we harnessed principles from coding theory, constrained systems, and calculus to address the challenge of DNA molecule fragmentation into non-overlapping substrings.

For studying the upper bounds on achievable rates of corresponding channels and conducting cardinality analyses of the presented codes, we employed techniques from combinatorics and calculus. Initially, we examined code construction for the probabilistic torn-paper channel, as described in [66], and assessed its performance under adversarial conditions. Utilizing tools such as the Lov'asz local lemma [67] and other combinatorial methods (similarly to techniques used independently in [85] and [22]) we demonstrated that this construction is suboptimal in adversarial scenarios.

Subsequently, we developed encoding and decoding algorithms for various related channels. We presented efficient constructions for different setups, encompassing noiseless channels, two versions of noisy channels, and a multi-stranded channel. Our primary strategy involved interleaving input strings with intelligently generated indices and synchronization markers. Leveraging Gray codes for indices and run-length-limited (RLL) strings [40, 84] as fundamental building blocks ensured marker uniqueness. To mitigate noise, we incorporated error-correcting codes and burst-erasure-correcting codes into our constructions.

Research Methods Invoked in Chapter 6

In Chapter 6, we developed several constructions for almost-balanced sequences. Our point of departure is our previous work [38], in which the constraint of *no windows with small period* was studied. Diverging from conventional methods (see e.g., [65, 72, 73]), our encoding algorithm in [38] breaks away from *monotonic progression* during input encoding. Instead, efficiency and convergence stem from a novel reduction to a graph problem. Drawing inspiration from this innovative technique, our presented work proposes an iterative approach for encoding sequences into almost-balanced forms, eliminating the need for progress between algorithm steps.

In conjunction with this approach, we introduced a modified version of arithmetic coding [60], a well-established lossless compression algorithm tailored to suit our constructions. To validate the correctness of our modified arithmetic encoder within our

encoding framework, we demonstrated its capability to compress any non-balanced input using tools from calculus and arithmetic. Additionally, tools from calculus were utilized to analyze the obtained codes.

Research Methods Invoked in Chapter 7

Our theoretical model in Chapter 7 draws inspiration from various proof-of-concept demonstrations in the field, including innovative approaches like DNA punch cards [69] and DNA composites [3, 14, 54]. Leveraging insights from these alternative DNA storage methods, we propose a new approach based on enzymatic DNA labeling.

Despite enzymatic DNA labeling being a widely-used technique in biochemistry, molecular biology, biotechnology, medical science, and genomic research [12, 20, 36, 45, 52, 86], its application specifically for storage remains relatively unexplored.

In our exploration, we propose a theoretical model that introduces a novel approach to utilize enzymatic DNA labeling for information storage, distinct from the model discussed in [29]. While [29] focuses on labeling for storage, its proposed framework deviates significantly from ours. Nonetheless, certain insights from [29] have influenced our work and are integrated into our proposed methodology.

Within our analysis, we primarily focus on a setup where all labels have uniform lengths, leveraging the (d,k)-RLL constraint [47]. Furthermore, we incorporate concepts from repeat-free sequences [22] and non-overlapping codes [8,40,42,82] to form the basis of our encoding strategies and facilitate our analysis. Combinatorial techniques and calculus play a crucial role in our study, enabling the cardinality analysis of our codes and the development of upper bounds.

Research Methods Invoked in Chapter 8

Our solution in Chapter 8 presents an end-to-end method to the DNA information retrieval problem. As part of this approach, it was divided into several components, each with its own functionality: encoding of sequences prior to DNA synthesis, clustering of reads post-sequencing, reconstruction, and decoding back to the original data. In addition, the interplay and interface between the different components were also taken into consideration as part of a complete system-level, optimized solution. The solution combines coding theory and deep learning methods to create a holistic and coherent pipeline to encode and decode the DNA data.

Our coding scheme is a modular pipeline composed of several components, each addressing a different purpose: index encoding, diagonal column encoding, constrained code, and tensor-product (TP) code [79]. The complete encoding and decoding pipeline is designed to integrate these four components in an interleaving order that allows each of them to achieve its designed goal without hindering the others. Our diagonal column encoding is based on Reed-Solomon (RS) code, and our TP code is composed

of RS code and Bose-Chaudhuri-Hocquenghem (BCH) code [46]. The suggested constrained code is based on a block encoding technique.

Our clustering step was optimized for speed rather than accuracy, and hence, we adopted a naïve approach based on simple and fast binning of the reads based on their index. For the reconstruction task, we introduced DNAformer, a solution based on Deep Neural Network (DNN) and algorithmic tools. Due to the high costs of synthesis and sequencing, and the lack of available datasets, we generate simulated training data for DNN-based reconstruction using statistics from real-world experiments [11, 62]. This strategy offers negligible training costs while generalization to real-world settings, a known challenge when using simulated data.

Our DNN employs a combination of convolutions and transformers within a Siamese network architecture. We adopt the concept of early convolutions before a transformer block to improve training stability and performance [80]. To learn the required alignment for each read independently, we introduce an alignment module that uses an Xception [15] inspired architecture with depthwise separable convolutions and multiple kernel heads. To learn the correlations between the different reads in a cluster and prepare the data for the Transformer module we used NCI aligner layer that includes an embedding module whose architecture is similar to the alignment module. The transformer module is a multi-head transformer architecture, used with Multi-Layer Perceptron as feedforward layers [74]. Our DNN architecture also includes a fusion layer which is a vector of learnable parameters with a length of the required encoded sequence. This layer combines the predictions of the two branches into a single prediction before a softmax operator, which transforms this representation into probabilities. For training, we employ a combination of cross-entropy and consistency loss functions, optimizing model performance and adaptability.

In addition to the DNN, our DNAformer incorporates the Conditional Probability Logic (CPL) algorithm to handle clusters with low confidence from the DNN output. This algorithm predicts the encoded sequence directly from the cluster, bypassing DNN predictions and prior knowledge of error rates. The CPL algorithm employs dynamic programming to estimate errors in a cluster and constructs a directed acyclic graph based on these estimates. The algorithm then identifies the longest path in this graph, representing the sequence with maximum probability based on its estimations.

The incorporation between the DNN, the CPL, and the decoder is facilitated by the confidence filter mechanism. Our confidence filter mechanism enables decision-making regarding DNN output reliability by classifying clusters as erasures or directing them to the CPL algorithm based on confidence scores and cluster sizes. This mechanism allows us to balance accuracy and runtime considerations effectively.

Furthermore, to ensure robust performance under varying conditions, we introduce the concept of a safety margin. The safety margin, derived from the error-correcting capabilities of the decoder, serves as a metric describing the DNAformer's robustness under specific working conditions. We achieve the desired safety margin through two control parameters: the minimum cluster size and a confidence threshold. Adjusting

these parameters allows us to calibrate the DNAformer's performance to meet specific requirements and adapt to different levels of noise and uncertainty in the input data.

Research Methods Invoked in Chapter 9

In Chapter 9, we introduce and tackle the DNA coverage depth problem, seeking to minimize the number of reads needed for information retrieval from the storage system. This challenge is reminiscent of the coupon collector's, dixie cup, and urn problems [23,25,27,56], and some of our explorations draws on insights and methodologies from these domains.

Throughout our investigation, we explore the coverage depth under different error-correcting codes, employing advanced tools from probability theory, combinatorics, and calculus. Many of our derivations rely on results concerning the geometric distribution and harmonic numbers [64], particularly their series and integral representations.

By employing Lagrange multipliers, we demonstrate that for Maximum Distance Separable (MDS) codes, coverage depth is minimized if and only if the channel is uniform. To analyze the noisy channel, we adopt a technique introduced by Erdős and Rényi in [23]. Additionally, we utilize the Chernoff bound [13], the Kullback–Leibler divergence [19], and the Lambert W function [10, 17].

In our examination of the random access setup, we utilize tools such as the inclusion-exclusion principle, generating functions, recursive formulas, and the complete probability theorem. Furthermore, we incorporate insights from results concerning the digamma function [4].

Part I

Deletion and Insertion Errors

Chapter 3

On the Size of Balls and Anticodes of Small Diameter under the Fixed-Length Levenshtein Metric

Daniella Bar-Lev, Tuvi Eztion, and Eitan Yaakobi

Abstract

The rapid development of DNA storage has brought the deletion and insertion channel to the front line of research. When the number of deletions is equal to the number of insertions, the *Fixed Length Levenshtein* (FLL) metric is the right measure for the distance between two words of the same length. Similar to any other metric, the size of a ball is one of the most fundamental parameters. In this work, we consider the minimum, maximum, and average size of a ball with radius one, in the FLL metric. The related minimum and the maximum size of a maximal anticode with diameter one are also considered.

3.1 Introduction

Coding for DNA storage has attracted significant attention in the previous decade due to recent experiments and demonstrations of the viability of storing information in macromolecules [2, 4,9,12,14,15,27,34,37]. Given the trends in cost decreases of DNA synthesis and sequencing, it is estimated that already within this decade DNA storage may become a highly competitive archiving technology. However, DNA molecules induce error patterns that are fundamentally different from their digital counterparts [17,18,21,29]; This distinction results from the specific error behavior in DNA and it is well-known that errors in DNA are typically in the form of substitutions, insertions, and deletions, where most published studies report that deletions are the most prominent ones, depending upon the specific technology for synthesis and sequencing. Hence, due to its high relevance to the error model in DNA storage coding for insertion and deletion errors has received renewed interest recently; see e.g. [5–8, 10, 13, 16, 25, 26, 28, 32, 33,

35]. This paper takes one more step in advancing this study and its goal is to study the size of balls and anticodes when the number of insertions equals to the number of deletions.

If a word $\boldsymbol{x} \in \mathbb{Z}_q^n$ can be transferred to a word $\boldsymbol{y} \in \mathbb{Z}_q^n$ using t deletions and t insertions (and cannot be transferred using a smaller number of deletions and insertions), then their ***Fixed Length Levenshtein* (FLL) *distance*** is t, which is denoted by $d_\ell(\boldsymbol{x}, \boldsymbol{y}) = t$. It is relatively easy to verify that the FLL distance defines a metric. Let $G = (V, E)$ be a graph whose set of vertices $V = \mathbb{Z}_q^n$ and two vertices $\boldsymbol{x}, \boldsymbol{y} \in V$ are connected by an edge if $d_\ell(\boldsymbol{x}, \boldsymbol{y}) = 1$. This graph represents the FLL distance. Moreover, the FLL distance defines a ***graphic metric***, i.e., it is a metric and for each $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^n$, $d_\ell(\boldsymbol{x}, \boldsymbol{y}) = t$ if and only if the length of the shortest path between $\boldsymbol{x}$ and $\boldsymbol{y}$ in G is t.

One of the most fundamental parameters in any metric is the size of a ball with a given radius t centered at a word $\boldsymbol{x}$. There are many metrics, e.g. the Hamming metric, the Johnson metric, or the Lee metric, where the size of a ball does not depend on the word $\boldsymbol{x}$. This is not the case in the FLL metric. Moreover, the graph G has a complex structure and it makes it much more difficult to find the exact size of any ball and even the size of a ball with minimum size and the size of a ball with maximum size. In [30], a formula for the size of the ball with radius one, centered at a word $\boldsymbol{x}$, in the FLL metric was given. This formula depends on the number of runs in the word and the lengths of its alternating segments (an alternating segment is a substring of consecutive alternating symbols). Nevertheless, while it is easy to compute the minimum size of a ball, it is still difficult to determine from this formula what the maximum size of a ball is. In this paper, we find explicit expressions for the minimum and maximum sizes of a ball when the ball is of radius one. We also find the average size of a ball when the radius of the ball is one. Finally, we consider the related basic concept of anticode in the FLL metric, where an anticode with diameter D is a code where the distance between any two elements of the code is at most D. Note, that a ball with radius R has diameter $2R$ and hence it is an anticode with diameter $2R$. We find the maximum size and the minimum size of maximal anticodes with diameter one, where an anticode with diameter one is maximal if any addition of a word to it will increase its diameter.

This paper is the first one which considers a comprehensive discussion and exact computation on the balls with radius one and the anticodes with diameter one in the FLL metric. The rest of this paper is organized as follows. Section 3.2 introduces some basic concepts, presents some of the known results on the sizes of balls, presents some results on equivalence of codes correcting deletions and insertions, and finally introduce some observations required for our exposition. The minimum size of a ball of any given radius in the FLL metric over $\mathbb{Z}_q$ is discussed in Section 3.3. Section 3.4 is devoted for the discussion on the maximum size of a ball with radius one in the FLL metric over $\mathbb{Z}_q$. The analysis of non-binary sequences is discussed in Section 3.4.1. It appears that contrary to many other coding problems the binary case is much more difficult to analyze and it is discussed in Section 3.4.2. For the binary case, the sequence for which the maximum size is obtained is presented in Theorem 3.25 and the maximum size is given in Corollary 3.26. The average size of the FLL ball with radius one over $\mathbb{Z}_q$ is computed in Section 3.5 and proved in Theorem 3.39. In Section 3.6, we consider binary maximal anticodes with diameter one. The maximum size of such an anticode is discussed in Section 3.6.1 and Section 3.6.2 is devoted to the minimum size of such anticodes. The results can be generalized for the non-binary case, but since they are more complicated and especially

messy, they are omitted.

3.2 Definitions and Previous Results

In this section, we present the definitions and notations as well as several results that will be used throughout the paper.

For an integer $q \geq 2$, let $\mathbb{Z}_q$ denote the set of integers $\{0, 1, \ldots, q-1\}$ and for an integer $n \geq 0$, let $\mathbb{Z}_q^n$ be the set of all sequences (words) of length n over the alphabet $\mathbb{Z}_q$, let $\mathbb{Z}_q^* = \bigcup_{n=0}^{\infty} \mathbb{Z}_q^n$, and let $[n]$ denote the set of integers $\{1, 2, \ldots, n\}$. For two sequences $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^n$, the distance between $\boldsymbol{x}$ and $\boldsymbol{y}$, $d(\boldsymbol{x}, \boldsymbol{y})$, can be measured in various ways. When the type of errors is substitution, the *Hamming distance* is the most natural to be considered. The *Hamming weight* of a sequence $\boldsymbol{x} \in \mathbb{Z}_q^*$, denoted by $\mathrm{wt}(\boldsymbol{x})$, is equal to the number of nonzero coordinates in $\boldsymbol{x}$. The Hamming distance between two sequences $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^n$, denoted by $d_H(\boldsymbol{x}, \boldsymbol{y})$, is the number of coordinates in which $\boldsymbol{x}$ and $\boldsymbol{y}$ differ. In other words, $d_H(\boldsymbol{x}, \boldsymbol{y})$ is the number of symbol-substitution operations required to transform $\boldsymbol{x}$ into $\boldsymbol{y}$. The Hamming distance is well known to be a metric on $\mathbb{Z}_q^n$ (also referred as the *Hamming space*), as it satisfies the three conditions of a metric (i.e., coincidence, symmetry and the triangle inequality). Given a distance d on a space V, the *t-ball* centered at $\boldsymbol{x} \in V$ is the set $\{\boldsymbol{y} \ : \ d(\boldsymbol{x}, \boldsymbol{y}) \leq t\}$. The *t-sphere* centered at $\boldsymbol{x} \in V$ is the set $\{\boldsymbol{y} \ : \ d(\boldsymbol{x}, \boldsymbol{y}) = t\}$. A *code* $\mathcal{C} \subseteq V$ is a subset of words from V. A related concept is an *anticode* with diameter D which is a code in V for which the distance between any two elements is at most D. Clearly, a t-ball is an anticode whose diameter is at most $2t$. The *Hamming t-ball* centered at $\boldsymbol{x} \in \mathbb{Z}_q^n$ will be denoted by $\mathcal{H}_t(\boldsymbol{x})$. For $\boldsymbol{x} \in \mathbb{Z}_q^n$, the number of words in the Hamming t-ball is a function of n, q and t. The number of such words is

$$|\mathcal{H}_t(\boldsymbol{x})| = \sum_{i=0}^{t} \binom{n}{i} (q-1)^i. \tag{3.1}$$

For an integer t, $0 \leq t \leq n$, a sequence $\boldsymbol{y} \in \mathbb{Z}_q^{n-t}$ is a *t-subsequence* of $\boldsymbol{x} \in \mathbb{Z}_q^n$ if $\boldsymbol{y}$ can be obtained from $\boldsymbol{x}$ by deleting t symbols from $\boldsymbol{x}$. In other words, there exist $n-t$ indices $1 \leq i_1 < i_2 < \cdots < i_{n-t} \leq n$ such that $y_j = x_{i_j}$, for all $1 \leq j \leq n-t$. We say that $\boldsymbol{y}$ is a *subsequence* of $\boldsymbol{x}$ if $\boldsymbol{y}$ is a t-subsequence of $\boldsymbol{x}$ for some t. Similarly, a sequence $\boldsymbol{y} \in \mathbb{Z}_q^{n+t}$ is a *t-supersequence* of $\boldsymbol{x} \in \mathbb{Z}_q^n$ if $\boldsymbol{x}$ is a t-subsequence of $\boldsymbol{y}$ and $\boldsymbol{y}$ is a *supersequence* of $\boldsymbol{x}$ if $\boldsymbol{y}$ is a t-supersequence of $\boldsymbol{x}$ for some t.

Definition 3.1. *The* deletion t-sphere *centered at $\boldsymbol{x} \in \mathbb{Z}_q^n$, $\mathcal{D}_t(\boldsymbol{x}) \subseteq \mathbb{Z}_q^{n-t}$, is the set of all t-subsequences of $\boldsymbol{x}$. The size of the largest deletion t-sphere in $\mathbb{Z}_q^n$ is denoted by $D_q(n, t)$. The* insertion t-sphere *centered at $\boldsymbol{x} \in \mathbb{Z}_q^n$, $\mathcal{I}_t(\boldsymbol{x}) \subseteq \mathbb{Z}_q^{n+t}$, is the set of all t-supersequences of $\boldsymbol{x}$.*

Let $\boldsymbol{x} \in \mathbb{Z}_q^n$ be a sequence. The size of the insertion t-sphere $|\mathcal{I}_t(\boldsymbol{x})|$ does not depend on $\boldsymbol{x}$ for any $0 \leq t \leq n$. To be exact, it was shown by Levenshtein [22] that

$$|\mathcal{I}_t(\boldsymbol{x})| = \sum_{i=0}^{t} \binom{n+t}{i} (q-1)^i. \tag{3.2}$$

On the other hand, calculating the exact size of the deletion sphere is one of the more intriguing problems when studying codes for deletions. Unlike substitution balls and insertions spheres, not all deletion spheres are of the same size. That is, the size of the deletion sphere, $|\mathcal{D}_t(\boldsymbol{x})|$, depends on the choice of the sequence $\boldsymbol{x}$. Let $\{\sigma_1, \ldots, \sigma_q\}$ be the symbols of $\mathbb{Z}_q$ in some order and let $\boldsymbol{c}(n) = (c_1, c_2, \ldots, c_n)$ be a sequence in $\mathbb{Z}_q^n$ such that $c_i = \sigma_i$ for $1 \leq i \leq q$ and $c_i = c_{i-q}$ for $i > q$. It was shown in Hirschberg and Regnier [19] that $\boldsymbol{c}(n)$ has the largest deletion t-sphere and its size is given by

$$D_q(n,t) = |\mathcal{D}_t(\boldsymbol{c}(n))| = \sum_{i=0}^{t} \binom{n-t}{i} D_{q-1}(t, t-i)$$

In particular, $D_2(n,t) = \sum_{i=0}^{t} \binom{n-t}{i}$ and $D_3(n,t) = \sum_{i=0}^{t} \binom{n-t}{i} \sum_{j=0}^{t-i} \binom{i}{j}$. The value $D_2(n,t)$ also satisfies the following recursion

$$D_2(n,t) = D_2(n-1,t) + D_2(n-2,t-1),$$

where the values for the basic cases can be evaluated by $D_2(n,t) = \sum_{i=0}^{t} \binom{n-t}{i}$.

Definition 3.2. *A* run *is a maximal subsequence composed of consecutive identical symbols. For a sequence $\boldsymbol{x} \in \mathbb{Z}_q^n$, the number of runs in $\boldsymbol{x}$ is denoted by $\rho(\boldsymbol{x})$.*

Example 3.3. *If $\boldsymbol{x} = 0000000$ then $\rho(\boldsymbol{x}) = 1$ since $\boldsymbol{x}$ has a single run of length 7 and for $\boldsymbol{y} = 1120212$ we have that $\rho(\boldsymbol{y}) = 6$ since $\boldsymbol{y}$ has six runs, the first is of length two and the others are of length one.*

There are upper and lower bounds on the size of the deletion ball which depend on the number of runs in the sequence. Namely, it was shown by Levenshtein [22] that

$$\binom{\rho(\boldsymbol{x}) - t + 1}{t} \leq |\mathcal{D}_t(\boldsymbol{x})| \leq \binom{\rho(\boldsymbol{x}) + t - 1}{t}.$$

Later, the lower bound was improved in [19]:

$$\sum_{i=0}^{t} \binom{\rho(\boldsymbol{x}) - t}{i} \leq |\mathcal{D}_t(\boldsymbol{x})| \leq \binom{\rho(\boldsymbol{x}) + t - 1}{t}. \tag{3.3}$$

Several more results on this value which take into account the number of runs appear in [24].

The *Levenshtein distance* between two words $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^*$, denoted by $d_L(\boldsymbol{x}, \boldsymbol{y})$, is the minimum number of insertions and deletions required to transform $\boldsymbol{x}$ into $\boldsymbol{y}$. Similarly, for two sequences $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^*$, $d_E(\boldsymbol{x}, \boldsymbol{y})$ denotes the *edit* distance between $\boldsymbol{x}$ and $\boldsymbol{y}$, which is the minimum number of insertions, deletions, and substitutions required to transform $\boldsymbol{x}$ into $\boldsymbol{y}$.

Definition 3.4. *Let t, n be integers such that $0 \leq t \leq n$. For a sequence $\boldsymbol{x} \in \mathbb{Z}_q^n$, the Levenshtein t-ball centered at $\boldsymbol{x} \in \mathbb{Z}_q^n$, $\widehat{\mathcal{L}}_t(\boldsymbol{x})$, is defined by*

$$\widehat{\mathcal{L}}_t(\boldsymbol{x}) \triangleq \{\boldsymbol{y} \in \mathbb{Z}_q^* \,:\, d_L(\boldsymbol{x}, \boldsymbol{y}) \leq t\}.$$

In case $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^n$, for some integer n, the *Fixed Length Levenshtein* (FLL) *distance* between $\boldsymbol{x}$ and $\boldsymbol{y}$, $d_\ell(\boldsymbol{x}, \boldsymbol{y})$, is the smallest t for which there exists a t-subsequence $\boldsymbol{z} \in \mathbb{Z}_q^{n-t}$ of both $\boldsymbol{x}$ and $\boldsymbol{y}$, i.e.

$$d_\ell(\boldsymbol{x}, \boldsymbol{y}) = \min\{t' : \mathcal{D}_{t'}(\boldsymbol{x}) \cap \mathcal{D}_{t'}(\boldsymbol{y}) \neq \varnothing\} = \frac{d_L(\boldsymbol{x}, \boldsymbol{y})}{2}. \tag{3.4}$$

In other words, t is the smallest integer for which there exists $\boldsymbol{z} \in \mathbb{Z}_q^{n-t}$ such that $\boldsymbol{z} \in \mathcal{D}_t(\boldsymbol{x})$ and $\boldsymbol{y} \in \mathcal{I}_t(\boldsymbol{z})$. Note that if $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^n$ and $\boldsymbol{x}$ is obtained from $\boldsymbol{y}$ by t_1 deletions and t_2 insertions, then $t_1 = t_2$.

Definition 3.5. *Let n, t be integers such that $0 \leq t \leq n$. For a sequence $\boldsymbol{x} \in \mathbb{Z}_q^n$, the FLL t-ball centered at $\boldsymbol{x} \in \mathbb{Z}_q^n$, $\mathcal{L}_t(\boldsymbol{x}) \subseteq \mathbb{Z}_q^n$, is defined by*

$$\mathcal{L}_t(\boldsymbol{x}) \triangleq \{\boldsymbol{y} \in \mathbb{Z}_q^n \ : \ d_\ell(\boldsymbol{x}, \boldsymbol{y}) \leq t\}.$$

We say that a subsequence $\boldsymbol{x}_{[i,j]} \triangleq x_i x_{i+1} \cdots x_j$ is an *alternating segment* if $\boldsymbol{x}_{[i,j]}$ is a sequence of alternating distinct symbols $\sigma, \sigma' \in \mathbb{Z}_q$. Note that $\boldsymbol{x}_{[i,j]}$ is a *maximal alternating segment* if $\boldsymbol{x}_{[i,j]}$ is an alternating segment and $\boldsymbol{x}_{[i-1,j]}, \boldsymbol{x}_{[i,j+1]}$ are not. The number of maximal alternating segments of a sequence $\boldsymbol{x}$ will be denoted by $A(\boldsymbol{x})$.

Example 3.6. *If $\boldsymbol{x} = 0000000$ then $A(\boldsymbol{x}) = 7$ since $\boldsymbol{x}$ has seven maximal alternating segments, each of length one, and for $\boldsymbol{x} = 1120212$ we have that $A(\boldsymbol{x}) = 4$ and the maximal alternating segments are $1,\ 12,\ 202,\ 212$.*

The following formula to compute $|\mathcal{L}_1(\boldsymbol{x})|$ as a function of $\rho(\boldsymbol{x})$ and $A(\boldsymbol{x})$ was given in [30]

$$|\mathcal{L}_1(\boldsymbol{x})| = \rho(\boldsymbol{x}) \cdot (n(q-1)-1) + 2 - \sum_{i=1}^{A(\boldsymbol{x})} \frac{(s_i - 1)(s_i - 2)}{2}, \tag{3.5}$$

where s_i for $1 \leq i \leq A(\boldsymbol{x})$ denotes the length of the i-th maximal alternating segment of $\boldsymbol{x}$.

Note that $|\widehat{\mathcal{L}}_1(\boldsymbol{x})|$, $|\widehat{\mathcal{L}}_2(\boldsymbol{x})|$ can be deduced from equations (3.2), (3.3), (3.4), and (3.5), since

$$\begin{aligned}
\widehat{\mathcal{L}}_1(\boldsymbol{x}) &= \mathcal{D}_1(\boldsymbol{x}) \cup \mathcal{I}_1(\boldsymbol{x}) \cup \{\boldsymbol{x}\}, \\
\widehat{\mathcal{L}}_2(\boldsymbol{x}) &= \mathcal{L}_1(\boldsymbol{x}) \cup \mathcal{D}_2(\boldsymbol{x}) \cup \mathcal{I}_2(\boldsymbol{x}) \cup \mathcal{D}_1(\boldsymbol{x}) \cup \mathcal{I}_1(\boldsymbol{x}),
\end{aligned}$$

and the length of the sequences in each ball is different which implies that the sets in these unions are disjoint. However, not much is known about the size of the Levenshtein ball and the FLL ball for arbitrary n, t and $\boldsymbol{x} \in \mathbb{Z}_q^n$.

For $\boldsymbol{x} \in \mathbb{Z}_q^*$, let $|\boldsymbol{x}|$ denote the length of $\boldsymbol{x}$ and for a set of indices $I \subseteq [|\boldsymbol{x}|]$, let $\boldsymbol{x}_I$ denote the *projection* of $\boldsymbol{x}$ on the ordered indices of I, which is the subsequence of $\boldsymbol{x}$ received by the symbols in the entries of I. For a symbol $\sigma \in \mathbb{Z}_q$, σ^n denotes the sequence with n consecutive σ's.

A word $\boldsymbol{x}$ is called a *common supersequence* (*subsequence*) of some sequences $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau$ if $\boldsymbol{x}$ is a supersequence (subsequence) of each one of these t words. The set of all shortest common supersequences of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau \in \mathbb{Z}_q^*$ is denoted by $\mathcal{SCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau)$ and $\mathsf{SCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau)$ is the *length of the shortest common supersequence* (*SCS*) of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau$, that is,

$$\mathsf{SCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau) = \min_{\boldsymbol{x} \in \mathcal{SCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau)} \{|\boldsymbol{x}|\}.$$

Similarly, $\mathcal{LCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau)$ is the set of all longest common subsequences of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau$ and $\mathsf{LCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau)$ is the *length of the longest common subsequence* (*LCS*) of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau$, that is,

$$\mathsf{LCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau) \triangleq \max_{\boldsymbol{x} \in \mathcal{LCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau)} \{|\boldsymbol{x}|\}.$$

This definition implies the following well-known property.

Claim 3.7. *For $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{Z}_q^n$, $\mathcal{D}_t(\boldsymbol{x}_1) \cap \mathcal{D}_t(\boldsymbol{x}_2) = \varnothing$ if and only if $\mathsf{LCS}(\boldsymbol{x}_1, \boldsymbol{x}_2) < n - t$.*

Combining (3.4) and Claim 3.7 implies the following result.

Corollary 3.8. *If $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{Z}_q^n$ then*

$$\mathsf{LCS}(\boldsymbol{x}_1, \boldsymbol{x}_2) = n - d_\ell(\boldsymbol{x}_1, \boldsymbol{x}_2).$$

For two sequences $\boldsymbol{x} \in \mathbb{Z}_q^n$ and $\boldsymbol{y} \in \mathbb{Z}_q^m$, the value of $\mathsf{LCS}(\boldsymbol{x}, \boldsymbol{y})$ is given by the following recursive formula [20]

$$\mathsf{LCS}(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 0 & n = 0 \text{ or } m = 0 \\ 1 + \mathsf{LCS}(\boldsymbol{x}_{[1,n-1]}, \boldsymbol{y}_{[1,m-1]}) & x_n = y_m \\ \max\left\{\mathsf{LCS}(\boldsymbol{x}_{[1,n-1]}, \boldsymbol{y}), \mathsf{LCS}(\boldsymbol{x}, \boldsymbol{y}_{[1,m-1]})\right\} & \text{otherwise} \end{cases}. \tag{3.6}$$

A subset $\mathcal{C} \subseteq \mathbb{Z}_q^n$ is a *t-deletion-correcting code* (*t-insertion-correcting code*, respectively) if for any two distinct codewords $\boldsymbol{c}, \boldsymbol{c}' \in \mathcal{C}$ we have that $\mathcal{D}_t(\boldsymbol{c}) \cap \mathcal{D}_t(\boldsymbol{c}') = \varnothing$ ($\mathcal{I}_t(\boldsymbol{c}) \cap \mathcal{I}_t(\boldsymbol{c}') = \varnothing$, respectively). Similarly, $\mathcal{C}$ is called a *(t_1, t_2)-deletion-insertion-correcting code* if for any two distinct codewords $\boldsymbol{c}$ and $\boldsymbol{c}'$ of the code $\mathcal{C}$, we have that $\mathcal{DI}_{t_1,t_2}(\boldsymbol{c}) \cap \mathcal{DI}_{t_1,t_2}(\boldsymbol{c}') = \varnothing$, where $\mathcal{DI}_{t_1,t_2}(\boldsymbol{x})$ is the set of all words that can be obtained from $\boldsymbol{x}$ by t_1 deletions and t_2 insertions. Levenshtein [22] proved that $\mathcal{C}$ is a t-deletion-correcting code if and only if $\mathcal{C}$ is a t-insertion-correcting code and if and only if $\mathcal{C}$ is a (t_1, t_2)-deletion-insertion-correcting code for every t_1, t_2 such that $t_1 + t_2 \leq t$. A straightforward generalization is the following result [11].

Lemma 3.9. *For all $t_1, t_2 \in \mathbb{N}$, if $\mathcal{C} \subseteq \mathbb{Z}_q^n$ is a (t_1, t_2)-deletion-insertion-correcting code, then $\mathcal{C}$ is also a $(t_1 + t_2)$-deletion-correcting code.*

Corollary 3.10. *For $\mathcal{C} \subseteq \mathbb{Z}_q^n$, the following statements are equivalent.*

1) $\mathcal{C}$ is a (t_1, t_2)-deletion-insertion-correcting code.

2) $\mathcal{C}$ is a $(t_1 + t_2)$-deletion-correcting code.

3) $\mathcal{C}$ is a $(t_1 + t_2)$-insertion-correcting code.

4) $\mathcal{C}$ is a (t'_1, t'_2)-deletion-insertion-correcting code for any t'_1, t'_2 such that

$$t'_1 + t'_2 = t_1 + t_2.$$

We further extend this result in the next lemma.

Lemma 3.11. *A code $\mathcal{C} \in \mathbb{Z}_q^n$ is a $(2t+1)$-deletion-correcting code if and only if the following two conditions are satisfied*
• $\mathcal{C}$ is a (t,t)-deletion-insertion-correcting code
and also
• if exactly $t+1$ FLL errors (i.e., $t+1$ insertions and $t+1$ deletions) occurred, then $\mathcal{C}$ can detect these $t+1$ FLL errors.

Proof. If $\mathcal{C}$ is a $(2t+1)$-deletion-correcting code, then by definition for any $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{C}$ we have that

$$\mathcal{D}_{2t+1}(\boldsymbol{c}_1) \cap \mathcal{D}_{2t+1}(\boldsymbol{c}_2) = \varnothing.$$

Therefore, by Claim 3.7 for any two distinct codewords $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{C}$ we have that

$$\mathsf{LCS}(\boldsymbol{c}_1, \boldsymbol{c}_2) \leq n - (2t+1).$$

Hence, by Corollary 3.8, $d_\ell(\boldsymbol{c}_1, \boldsymbol{c}_2) \geq 2(t+1)$. Since the FLL metric is graphic, it follows that $\mathcal{C}$ can correct up to t FLL errors and if exactly $t+1$ FLL errors occurred it can detect them.

For the other direction, assume that $\mathcal{C}$ is a (t,t)-deletion-insertion-correcting code and if exactly $t+1$ FLL errors occurred, then $\mathcal{C}$ can detect them. By Lemma 3.9, $\mathcal{C}$ is a $(2t)$-deletion-correcting code which implies that $\mathcal{D}_{2t}(\boldsymbol{c}_1) \cap \mathcal{D}_{2t}(\boldsymbol{c}_2) = \varnothing$ for all $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{C}$, and hence by (3.4) we have that

$$\forall \boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{C}: \quad d_\ell(\boldsymbol{c}_1, \boldsymbol{c}_2) > 2t.$$

Let us assume to the contrary that there exist two codewords $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{C}$ such that $d_\ell(\boldsymbol{c}_1, \boldsymbol{c}_2) = 2t+1$. Since the FLL metric is a graphic metric, it follows that there exists a word $\boldsymbol{y} \in \mathbb{Z}_q^n$ such that $d_\ell(\boldsymbol{c}_1, \boldsymbol{y}) = t$ and $d_\ell(\boldsymbol{y}, \boldsymbol{c}_2) = t+1$. Hence, if the received word is $\boldsymbol{y}$, then the submitted codeword can be either $\boldsymbol{c}_1$ (t errors) or $\boldsymbol{c}_2$ ($t+1$ errors) which contradicts the fact that in $\mathcal{C}$ up to t FLL errors can be corrected and exactly $t+1$ FLL errors can be detected. Hence,

$$\forall \boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{C}: \quad d_\ell(\boldsymbol{c}_1, \boldsymbol{c}_2) > 2t+1,$$

and by definition, $\mathcal{C}$ can correct $2t+1$ deletions. □

3.3 The Minimum Size of an FLL Ball

In this section, the explicit expression for the minimum size of an FLL t-ball of any radius t is derived. Although this result is rather simple and straightforward, it is presented here for

the completeness of the problems studied in the paper. Since changing the symbol in the i-th position from σ to σ' in any sequence $\boldsymbol{x}$ can be done by first deleting σ in the i-th position of $\boldsymbol{x}$ and then inserting σ' in the same position of $\boldsymbol{x}$, it follows that

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^n : \quad d_H(\boldsymbol{x}, \boldsymbol{y}) \geq d_\ell(\boldsymbol{x}, \boldsymbol{y}).$$

Since $\boldsymbol{y} \in \mathcal{H}_t(\boldsymbol{x})$ if and only if $d_H(\boldsymbol{x}, \boldsymbol{y}) \leq t$ and $\boldsymbol{y} \in \mathcal{L}_t(\boldsymbol{x})$ if and only if $d_\ell(\boldsymbol{x}, \boldsymbol{y}) \leq t$, the following results are immediately implied.

Lemma 3.12. *If $n \geq t \geq 0$ are integers and $\boldsymbol{x} \in \mathbb{Z}_q^n$, then $\mathcal{H}_t(\boldsymbol{x}) \subseteq \mathcal{L}_t(\boldsymbol{x})$.*

Corollary 3.13. *For any two integers $n \geq t \geq 0$ and any sequence $\boldsymbol{x} \in \mathbb{Z}_q^n$, $|\mathcal{H}_t(\boldsymbol{x})| \leq |\mathcal{L}_t(\boldsymbol{x})|$.*

Lemma 3.14. *If $n > t \geq 0$ are integers, then $\mathcal{H}_t(\boldsymbol{x}) = \mathcal{L}_t(\boldsymbol{x})$ if and only if $\boldsymbol{x} = \sigma^n$ for $\sigma \in \mathbb{Z}_q$.*

Proof. Assume first w.l.o.g. that $\boldsymbol{x} = 0^n$ and let $\boldsymbol{y} \in \mathcal{L}_t(\boldsymbol{x})$ be a sequence obtained from $\boldsymbol{x}$ by at most t insertions and t deletions. Hence, $\mathrm{wt}(\boldsymbol{y}) \leq t$ and $\boldsymbol{y} \in \mathcal{H}_t(\boldsymbol{x})$, which implies that $\mathcal{L}_t(\boldsymbol{x}) \subseteq \mathcal{H}_t(\boldsymbol{x})$. Therefore, Lemma 3.12 implies that $\mathcal{H}_t(\boldsymbol{x}) = \mathcal{L}_t(\boldsymbol{x})$.

For the other direction, assume that $\mathcal{H}_t(\boldsymbol{x}) = \mathcal{L}_t(\boldsymbol{x})$ and let $\boldsymbol{x} \in \mathbb{Z}_q^n$ where $\boldsymbol{x} \neq \sigma^n$ for all $\sigma \in \mathbb{Z}_q$. Since by Lemma 3.12, $\mathcal{H}_t(\boldsymbol{x}) \subseteq \mathcal{L}_t(\boldsymbol{x})$, to complete the proof, it is sufficient to show that there exists a sequence $\boldsymbol{y} \in \mathcal{L}_t(\boldsymbol{x}) \backslash \mathcal{H}_t(\boldsymbol{x})$. Denote $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and let i be the smallest index for which $x_i \neq x_{i+1}$. Let $\boldsymbol{y}$ be the sequence defined by

$$\boldsymbol{y} \triangleq (y_1, y_2, \ldots, y_{i-1}, x_{i+1}, x_i, y_{i+2}, \ldots, y_n),$$

where $y_j \neq x_j$ for the first $t-1$ indices (for which $j \notin \{i, i+1\}$) and $y_j = x_j$ otherwise. Clearly, $\boldsymbol{y}$ differs from $\boldsymbol{x}$ in $t+1$ indices and therefore $\boldsymbol{y} \notin \mathcal{H}_t(\boldsymbol{x})$. On the other hand, $\boldsymbol{y}$ can be obtained from $\boldsymbol{x}$ by first deleting x_i and inserting it to the right of x_{i+1} and then applying $t-1$ deletions and $t-1$ insertions whenever $y_j \neq x_j$ (where $j \notin \{i, i+1\}$). Thus, $\boldsymbol{y} \in \mathcal{L}_t(\boldsymbol{x}) \backslash \mathcal{H}_t(\boldsymbol{x})$ which completes the proof. □

The following simple corollary is a direct result of Corollary 3.13, Lemma 3.14 and (3.1).

Corollary 3.15. *If $n > t \geq 0$ and $q > 1$ are integers, then the size of the minimum FLL t-ball is*

$$\min_{\boldsymbol{x} \in \mathbb{Z}_q^n} |\mathcal{L}_t(\boldsymbol{x})| = \sum_{i=0}^{t} \binom{n}{i} (q-1)^i,$$

and the minimum is obtained only by the balls centered at $\boldsymbol{x} = \sigma^n$ for any $\sigma \in \mathbb{Z}_q$.

3.4 The Maximum FLL Balls with Radius One

The goal of this section is to compute the size of a ball with maximum size and its centre. For this purpose it is required first to compute the size of a ball. The size of the FLL 1-ball centered at $\boldsymbol{x} \in \mathbb{Z}_q^n$ was proved in [30] and given in (3.5). In the analysis of the maximum ball

we distinguish between the binary case and the non-binary case. Surprisingly, the computation of the non-binary case is not a generalization of the binary case. That is, the binary case is not a special case of the non-binary case. Even more surprising is that the analysis of the non-binary case is much simpler than the analysis of the binary case. Hence, we start with the analysis of the non-binary case which is relatively simple.

3.4.1 The Non-Binary Case

By (3.5), the size of a ball with radius one centered at $\boldsymbol{x}$ depends on $\rho(\boldsymbol{x})$, the number of runs in $\boldsymbol{x}$. For a given number of runs $1 \leq r \leq n$, the size of a ball depends on the lengths of the maximal alternating segments in $\boldsymbol{x}$. The following lemma is an immediate consequence of (3.5).

Lemma 3.16. *If $n > 0$ and $1 \leq r \leq n$, then*

$$\arg\max_{\substack{\boldsymbol{x}\in\mathbb{Z}_q^n \\ \rho(\boldsymbol{x})=r}} |\mathcal{L}_1(\boldsymbol{x})| = \arg\min_{\substack{\boldsymbol{x}\in\mathbb{Z}_q^n \\ \rho(\boldsymbol{x})=r}} \left\{ \sum_{i=1}^{A(\boldsymbol{x})} \frac{(s_i-1)(s_i-2)}{2} \right\}.$$

Proof. Let $\boldsymbol{x} \in \mathbb{Z}_q^n$ be a sequence with exactly r runs. Since $r(n(q-1)-1)+2$ is a constant and

$$\sum_{i=1}^{A(\boldsymbol{x})} \frac{(s_i-1)(s_i-2)}{2} \geq 0,$$

the claim follows immediately from (3.5). □

Corollary 3.17. *If $n > 0$ and $1 \leq r \leq n$, then*

$$\max_{\substack{\boldsymbol{x}\in\mathbb{Z}_q^n \\ \rho(\boldsymbol{x})=r}} |\mathcal{L}_1(\boldsymbol{x})| = r(n(q-1)-1)+2 - \min_{\substack{\boldsymbol{x}\in\mathbb{Z}_q^n \\ \rho(\boldsymbol{x})=r}} \left\{ \sum_{i=1}^{A(\boldsymbol{x})} \frac{(s_i-1)(s_i-2)}{2} \right\}.$$

Note that

$$\sum_{i=1}^{A(\boldsymbol{x})} \frac{(s_i-1)(s_i-2)}{2} = 0 \iff \forall 1 \leq i \leq A(\boldsymbol{x}) : s_i \in \{1,2\}. \tag{3.7}$$

The following claim is a straightforward result from the definitions of a run and an alternating segment.

Lemma 3.18. *Let $n > 0$ and let $\boldsymbol{x} \in \mathbb{Z}_q^n$. For $1 \leq i \leq \rho(\boldsymbol{x})$, denote by r_i the length of the i-th run and by $\sigma_i \in \mathbb{Z}_q$ the symbol of the i-th run. Then all the maximal alternating segments of $\boldsymbol{x}$ have lengths at most two ($s_i \leq 2$ for each i) if and only if for each $1 \leq i \leq \rho(\boldsymbol{x})-2$, $\sigma_i \neq \sigma_{i+2}$ or $r_{i+1} > 1$.*

The maximum value of $|\mathcal{L}_1(\boldsymbol{x})|$ for non-binary alphabet was given in [31] without a proof. For $q = 2$ the value of $|\mathcal{L}_1(\boldsymbol{x})|$ given in [31] without a proof is not accurate and we will give the exact value with a complete proof.

Theorem 3.19. *For $q > 2$, the maximum FLL 1-balls are the balls centered at $\boldsymbol{x} \in \mathbb{Z}_q^n$, such that the number of runs in $\boldsymbol{x}$ is n (i.e., any two consecutive symbols are different) and $x_i \neq x_{i+2}$ for all $1 \leq i \leq n-2$. In addition, the maximum size of an FLL 1-ball is,*

$$\max_{\boldsymbol{x} \in \mathbb{Z}_q^n} |\mathcal{L}_1(\boldsymbol{x})| = n^2(q-1) - n + 2.$$

Proof. Corollary 3.17 implies that

$$\begin{aligned}\max_{\boldsymbol{x} \in \mathbb{Z}_q^n} |\mathcal{L}_1(\boldsymbol{x})| &= \max_{r \in \{1,\ldots,n\}} \left\{ \max_{\substack{\boldsymbol{x} \in \mathbb{Z}_q^n \\ \rho(\boldsymbol{x})=r}} |\mathcal{L}_1(\boldsymbol{x})| \right\} \\ &= \max_{r \in \{1,\ldots,n\}} \left\{ r(n(q-1)-1) + 2 - \min_{\substack{\boldsymbol{x} \in \mathbb{Z}_q^n \\ \rho(\boldsymbol{x})=r}} \left\{ \sum_{i=1}^{A(\boldsymbol{x})} \frac{(s_i-1)(s_i-2)}{2} \right\} \right\}.\end{aligned}$$

Clearly, $r(n(q-1)-1)+2$ is maximized for $r = n$ and therefore, using (3.7), we conclude that $\max_{\boldsymbol{x} \in \mathbb{Z}_q^n} |\mathcal{L}_1(\boldsymbol{x})|$ can be obtained for each $\boldsymbol{x} \in \mathbb{Z}_q^n$ such that $\rho(\boldsymbol{x}) = n$ and $s_i \leq 2$ for each i. Note that $\sigma_i = x_i$ since $r = n$. By Lemma 3.18, it implies that $x_i \neq x_{i+2}$ or $r_{i+1} > 1$ for each $1 \leq i \leq n-2$. Since $q > 2$, it follows that there exists such an assignment for the symbols of each run such that $x_i \neq x_{i+2}$ for each $1 \leq i \leq r-2$. It follows that

$$\max_{\boldsymbol{x} \in \mathbb{Z}_q^n} |\mathcal{L}_1(\boldsymbol{x})| = n^2(q-1) - n + 2.$$

□

3.4.2 The Binary Case

The analysis to find the maximum ball for binary sequences is more difficult, since by definition of a run, there is no sequence $\boldsymbol{x}$ with n runs such that $x_i \neq x_{i+2}$ (see Theorem 3.19) for some i. Note also that since in the binary case two maximal alternating segments cannot overlap it holds that $\sum_{i=1}^{A(\boldsymbol{x})} s_i = n$ for any binary sequence $\boldsymbol{x}$.

For a sequence $\boldsymbol{x} \in \mathbb{Z}_2^n$, the *alternating segments profile* of $\boldsymbol{x}$ is $(s_1, s_2, \ldots, s_{A(\boldsymbol{x})})$. Note that each alternating segments profile defines exactly two binary sequences.

Lemma 3.20. *If $\boldsymbol{x} \in \mathbb{Z}_2^n$ then $\rho(\boldsymbol{x}) = n + 1 - A(\boldsymbol{x})$.*

Proof. Let $\boldsymbol{x} \in \mathbb{Z}_2^n$ be a sequence and let $\boldsymbol{x}_{[i,j]}$ and $\boldsymbol{x}_{[i',j']}$ be two consecutive maximal alternating segments such that $i < i'$. Since $\boldsymbol{x}$ is a binary sequence, it follows that two maximal alternating segments cannot overlap, and hence $i' = j + 1$. Now, let $\alpha = A(\boldsymbol{x})$ and we continue to prove the claim of the lemma by induction on α for any given $n \geq 1$. For $\alpha = 1$, there is one maximal alternating segment whose length is clearly n which consists of alternating symbols, i.e., there are $\rho(\boldsymbol{x}) = n$ runs as required. Assume the claim holds for any α' such that $1 \leq \alpha' < \alpha$ and let $\boldsymbol{x} \in \mathbb{Z}_2^n$ be a sequence with exactly α maximal alternating segments. Denote by $\boldsymbol{x}'$ the sequence that is obtained from $\boldsymbol{x}$ by deleting its last maximal alternating segment $\boldsymbol{x}''$. By the induction hypothesis

$$\rho(\boldsymbol{x}') = (n - s_\alpha) + 1 - (\alpha - 1) = n + 2 - s_\alpha - \alpha,$$

where s_α is the length of $\boldsymbol{x}''$. Clearly, the first symbol of $\boldsymbol{x}''$ is equal to the last symbol in $\boldsymbol{x}'$. Thus,

$$\begin{aligned}\rho(\boldsymbol{x}) &= \rho(\boldsymbol{x}'\boldsymbol{x}'') = \rho(\boldsymbol{x}') + s_\alpha - 1 \\ &= n + 2 - s_\alpha - \alpha + s_\alpha - 1 = n + 1 - \alpha.\end{aligned}$$

□

Notice that $\rho(\boldsymbol{x}) = n + 1 - A(\boldsymbol{x})$ does not hold for alphabet size $q > 2$. To clarify, consider the sequences $\boldsymbol{x}_1 = 0120$, $\boldsymbol{x}_2 = 0101$ and $\boldsymbol{x}_3 = 0102$, each of the sequences has four runs even though they differ in the number of maximal alternating segments; $A(\boldsymbol{x}_1) = 3$, $A(\boldsymbol{x}_2) = 1$ and $A(\boldsymbol{x}_3) = 2$.

Definition 3.21. *For a positive integer α, $\boldsymbol{x}^{(\alpha)} \in \mathbb{Z}_2^n$ is an α-**balanced sequence** if $A(\boldsymbol{x}) = \alpha$ and $s_i \in \{\lceil \frac{n}{\alpha} \rceil, \lceil \frac{n}{\alpha} \rceil - 1\}$ for all $i \in \{1, \ldots, \alpha\}$.*

Lemma 3.22. *If n is a positive integer and $\alpha \in \{1, \ldots, n\}$ then*

$$\arg\max_{\substack{\boldsymbol{x} \in \mathbb{Z}_2^n \\ A(\boldsymbol{x}) = \alpha}} |\mathcal{L}_1(\boldsymbol{x})| = \{\boldsymbol{x} \in \mathbb{Z}_2^n : \boldsymbol{x} \text{ is an } \alpha\text{-balanced sequence}\}.$$

Proof. For a sequence $\boldsymbol{x} \in \mathbb{Z}_2^n$ such that $A(\boldsymbol{x}) = \alpha$, Lemma 3.20 implies that $\rho(\boldsymbol{x}) = n + 1 - \alpha$. Hence, by Lemma 3.16,

$$\begin{aligned}\arg\max_{\substack{\boldsymbol{x} \in \mathbb{Z}_2^n \\ A(\boldsymbol{x}) = \alpha}} |\mathcal{L}_1(\boldsymbol{x})| &= \arg\min_{\substack{\boldsymbol{x} \in \mathbb{Z}_2^n \\ A(\boldsymbol{x}) = \alpha}} \sum_{i=1}^{\alpha} \frac{(s_i - 1)(s_i - 2)}{2} \\ &= \arg\min_{\substack{\boldsymbol{x} \in \mathbb{Z}_2^n \\ A(\boldsymbol{x}) = \alpha}} \sum_{i=1}^{\alpha} (s_i^2 - 3s_i + 2) \\ &= \arg\min_{\substack{\boldsymbol{x} \in \mathbb{Z}_2^n \\ A(\boldsymbol{x}) = \alpha}} \left(\sum_{i=1}^{\alpha} s_i^2 - 3 \sum_{i=1}^{\alpha} s_i + 2\alpha \right) \\ &\overset{(a)}{=} \arg\min_{\substack{\boldsymbol{x} \in \mathbb{Z}_2^n \\ A(\boldsymbol{x}) = \alpha}} \left(\sum_{i=1}^{\alpha} s_i^2 - 3n + 2\alpha \right) \\ &= \arg\min_{\substack{\boldsymbol{x} \in \mathbb{Z}_2^n \\ A(\boldsymbol{x}) = \alpha}} \sum_{i=1}^{\alpha} s_i^2,\end{aligned}$$

where (a) holds since alternating segments cannot overlap for binary sequences and therefore $\sum_{i=1}^{\alpha} s_i = n$.

Assume $\boldsymbol{x} \in \mathbb{Z}_2^n$ is a sequence such that $A(\boldsymbol{x}) = \alpha$, $(s_1, \ldots, s_\alpha)$ is the alternating segments profile of $\boldsymbol{x}$ and $\sum_{i=1}^{\alpha} s_i^2$ is minimal among all sequences in $\mathbb{Z}_2^n$. Assume to the contrary that

$\boldsymbol{x}$ is not an α-balanced sequence. Then there exist indices $i \neq j$ such that $s_i \leq \left\lceil \frac{n}{\alpha} \right\rceil - 1$ and $s_j > \left\lceil \frac{n}{\alpha} \right\rceil$ or there exist indices $i \neq j$ such that $s_i < \left\lceil \frac{n}{\alpha} \right\rceil - 1$ and $s_j \geq \left\lceil \frac{n}{\alpha} \right\rceil$. Consider a sequence $\boldsymbol{x}'$ with the alternating segments profile $(\nu_1, \ldots, \nu_\alpha)$ where

$$\nu_k = \begin{cases} s_i + 1 & \text{if } k = i \\ s_j - 1 & \text{if } k = j \\ s_k & \text{otherwise.} \end{cases}$$

Therefore,

$$\begin{aligned} \sum_{k=1}^{\alpha} \nu_k^2 - \sum_{k=1}^{\alpha} s_k^2 &= \sum_{k=1}^{\alpha} \left(\nu_k^2 - s_k^2 \right) = (\nu_i^2 - s_i^2) + (\nu_j^2 - s_j^2) \\ &= \left((s_i + 1)^2 - s_i^2 \right) + \left((s_j - 1)^2 - s_j^2 \right) \\ &= \left(s_i^2 + 2s_i + 1 - s_i^2 \right) + \left(s_j^2 - 2s_j + 1 - s_j^2 \right) \\ &= 2(s_i - s_j + 1) \\ &< 2\left(\left\lceil \frac{n}{\alpha} \right\rceil - 1 - \left\lceil \frac{n}{\alpha} \right\rceil + 1 \right) = 0, \end{aligned}$$

and hence $\sum_{k=1}^{\alpha} \nu_k^2 < \sum_{k=1}^{\alpha} s_k^2$. This implies that if $\boldsymbol{x}$ is not an α-balanced sequence, then $\sum_{k=1}^{\alpha} s_k^2$ is not minimal, a contradiction. Thus,

$$\arg\max_{\substack{\boldsymbol{x} \in \mathbb{Z}_2^n \\ A(\boldsymbol{x}) = \alpha}} |\mathcal{L}_1(\boldsymbol{x})| = \arg\min_{\substack{\boldsymbol{x} \in \mathbb{Z}_2^n \\ A(\boldsymbol{x}) = \alpha}} \sum_{i=1}^{\alpha} s_i^2 = \{\boldsymbol{x} \in \mathbb{Z}_2^n \ : \ \boldsymbol{x} \text{ is an } \alpha\text{-balanced sequence}\}.$$

□

Lemma 3.23. *Let $\boldsymbol{x}^{(\alpha)}$ be an α-balanced sequence of length n. Then,*

$$\begin{aligned} \left| \mathcal{L}_1\left(\boldsymbol{x}^{(\alpha)} \right) \right| &= (n + 1 - \alpha)(n - 1) + 2 \\ &\quad - \frac{k}{2}\left(\left\lceil \frac{n}{\alpha} \right\rceil - 1 \right)\left(\left\lceil \frac{n}{\alpha} \right\rceil - 2 \right) \\ &\quad - \frac{\alpha - k}{2}\left(\left\lceil \frac{n}{\alpha} \right\rceil - 2 \right)\left(\left\lceil \frac{n}{\alpha} \right\rceil - 3 \right), \end{aligned}$$

where $k \equiv n \pmod{\alpha}$ and $1 \leq k \leq \alpha$.

Proof. By (3.5) we have that

$$\left| \mathcal{L}_1\left(\boldsymbol{x}^{(\alpha)} \right) \right| = \rho\left(\boldsymbol{x}^{(\alpha)} \right) \cdot (n - 1) + 2 - \sum_{i=1}^{\alpha} \frac{(s_i - 1)(s_i - 2)}{2}, \tag{3.8}$$

and Lemma 3.20 implies that $\rho\left(\boldsymbol{x}^{(\alpha)} \right) = n + 1 - \alpha$. Let k be the number of entries in the alternating segments profile of $\boldsymbol{x}^{(\alpha)}$ such that $s_i = \left\lceil \frac{n}{\alpha} \right\rceil$. Note further that $\sum_{i=1}^{\alpha} s_i = n$ and $s_i \in \left\{ \left\lceil \frac{n}{\alpha} \right\rceil, \left\lceil \frac{n}{\alpha} \right\rceil - 1 \right\}$ for $1 \leq i \leq \alpha$. Hence,

$$k\left\lceil \frac{n}{\alpha} \right\rceil + (\alpha - k)\left(\left\lceil \frac{n}{\alpha} \right\rceil - 1 \right) = n,$$

which is equivalent to

$$k = n - \alpha\Big(\Big\lceil \frac{n}{\alpha} \Big\rceil - 1\Big).$$

Therefore, k is the value between 1 to α such that $k \equiv n \pmod{\alpha}$. Thus, by (3.8) we have that

$$\begin{aligned}\Big|\mathcal{L}_1\Big(\boldsymbol{x}^{(\alpha)}\Big)\Big| &= (n+1-\alpha)(n-1)+2 \\ &\quad - \frac{k}{2}\Big(\Big\lceil \frac{n}{\alpha} \Big\rceil - 1\Big)\Big(\Big\lceil \frac{n}{\alpha} \Big\rceil - 2\Big) \\ &\quad - \frac{\alpha-k}{2}\Big(\Big\lceil \frac{n}{\alpha} \Big\rceil - 2\Big)\Big(\Big\lceil \frac{n}{\alpha} \Big\rceil - 3\Big).\end{aligned}$$

□

By Lemma 3.22 we have that

$$\begin{aligned}\max_{x\in\mathbb{Z}_2^n} |\mathcal{L}_1(\boldsymbol{x})| &= \max_{1\le\alpha\le n}\left\{ \max_{\substack{\boldsymbol{x}\in\mathbb{Z}_2^n \\ A(\boldsymbol{x})=\alpha}} |\mathcal{L}_1(\boldsymbol{x})| \right\} \\ &= \max_{1\le\alpha\le n}\Big\{\Big|\mathcal{L}_1\Big(\boldsymbol{x}^{(\alpha)}\Big)\Big|\Big\},\end{aligned}$$

and the size $\Big|\mathcal{L}_1\Big(\boldsymbol{x}^{(\alpha)}\Big)\Big|$ for $1 \le \alpha \le n$ is given in Lemma 3.23. Hence, our goal is to find the set

$$\mathsf{A} \triangleq \arg\max_{1\le\alpha\le n}\Big\{\Big|\mathcal{L}_1\Big(\boldsymbol{x}^{(\alpha)}\Big)\Big|\Big\},$$

i.e., for which values of α the maximum of $|\mathcal{L}_1\Big(\boldsymbol{x}^{(\alpha)}\Big)|$ is obtained. The answer for this question is given in the following lemma whose proof can be found in the Appendix.

Lemma 3.24. *Let $\boldsymbol{x}^{(\alpha)}$ be an α-balanced sequence of length $n > 1$. Then,*

$$\Big|\mathcal{L}_1\Big(\boldsymbol{x}^{(\alpha)}\Big)\Big| > \Big|\mathcal{L}_1\Big(\boldsymbol{x}^{(\alpha-1)}\Big)\Big|$$

if and only if $n > 2(\alpha-1)\alpha$.

Theorem 3.25. *If n is an integer, then*

$$\mathsf{A} = \arg\min_{\alpha\in\mathbb{N}}\left\{\left|\alpha - \frac{1}{2}\sqrt{1+2n}\right|\right\},$$

and the maximum FLL 1-balls are the balls centered at the α-balanced sequences of length n, for $\alpha \in \mathsf{A}$. In addition, the size of the maximum FLL 1-balls is given by

$$\begin{aligned}\max_{\boldsymbol{x}\in\mathbb{Z}_2^n}\{|\mathcal{L}_1(\boldsymbol{x})|\} &= n^2 - n\alpha + \alpha + 1 \\ &\quad - \frac{k}{2}\Big(\Big\lceil \frac{n}{\alpha} \Big\rceil - 1\Big)\Big(\Big\lceil \frac{n}{\alpha} \Big\rceil - 2\Big) \\ &\quad - \frac{\alpha-k}{2}\Big(\Big\lceil \frac{n}{\alpha} \Big\rceil - 2\Big)\Big(\Big\lceil \frac{n}{\alpha} \Big\rceil - 3\Big),\end{aligned}$$

where $k \equiv n \pmod{\alpha}$ *and* $1 \leq k \leq \alpha$.

Proof. Let n be a positive integer. By Lemma 3.22 we have that

$$\begin{aligned}\max_{x\in\mathbb{Z}_2^n} |\mathcal{L}_1(\boldsymbol{x})| &= \max_{1\leq\alpha\leq n}\left\{\max_{\substack{\boldsymbol{x}\in\mathbb{Z}_2^n\\ A(\boldsymbol{x})=\alpha}} |\mathcal{L}_1(\boldsymbol{x})|\right\}\\ &= \max_{1\leq\alpha\leq n}\left\{\left|\mathcal{L}_1\left(\boldsymbol{x}^{(\alpha)}\right)\right|\right\}.\end{aligned}$$

If there exists an integer α, $1 \leq \alpha \leq n$ such that $n = 2(\alpha-1)\alpha$, then by Lemma 3.23,

$$\left|\mathcal{L}_1\left(\boldsymbol{x}^{(\alpha)}\right)\right| = \left|\mathcal{L}_1\left(\boldsymbol{x}^{(\alpha-1)}\right)\right|.$$

Additionally, by Lemma 3.24, for $n > 2(\alpha-1)\alpha$ we have that

$$\left|\mathcal{L}_1\left(\boldsymbol{x}^{(\alpha)}\right)\right| > \left|\mathcal{L}_1\left(\boldsymbol{x}^{(\alpha-1)}\right)\right|,$$

which implies that $\left|\mathcal{L}_1\left(\boldsymbol{x}^{(\alpha)}\right)\right|$ is maximized for $\alpha \in \{1,\ldots,n\}$ such that

$$2\alpha(\alpha+1) \geq n \geq 2(\alpha-1)\alpha. \tag{3.9}$$

To find α we have to solve the two quadratic equations from (3.9). The solution for α must satisfies both equations and hence $-\frac{1}{2} + \frac{\sqrt{1+2n}}{2} \leq \alpha \leq \frac{1}{2} + \frac{\sqrt{1+2n}}{2}$. Namely, for $\alpha \in \mathsf{A}$,

$$\max_{\boldsymbol{x}\in\mathbb{Z}_2^n}\{|\mathcal{L}_1(\boldsymbol{x})|\} = \left|\mathcal{L}_1\left(\boldsymbol{x}^{(\alpha)}\right)\right|$$

The size of $\mathcal{L}_1\left(\boldsymbol{x}^{(\alpha)}\right)$ was derived in Lemma 3.23, which completes the proof. □

Corollary 3.26. *If* n *is an integer, then*

$$\max_{\boldsymbol{x}\in\mathbb{Z}_2^n}\{|\mathcal{L}_1(\boldsymbol{x})|\} = n^2 - \sqrt{2}n^{\frac{3}{2}} + O(n).$$

Proof. By Theorem 3.25 we have that $\max_{x\in\mathbb{Z}_2^n}\{|\mathcal{L}_1(\boldsymbol{x})|\} = \left|\mathcal{L}_1\left(\boldsymbol{x}^{(\alpha)}\right)\right|$ for $\alpha = \left[\frac{1}{2}\sqrt{1+2n}\right]$. By Lemma 3.23 we have that

$$\begin{aligned}\left|\mathcal{L}_1\left(\boldsymbol{x}^{(\alpha)}\right)\right| &= (n+1-\alpha)(n-1)+2\\ &\quad - \frac{k}{2}\left(\left\lceil\frac{n}{\alpha}\right\rceil - 1\right)\left(\left\lceil\frac{n}{\alpha}\right\rceil - 2\right)\\ &\quad - \frac{\alpha-k}{2}\left(\left\lceil\frac{n}{\alpha}\right\rceil - 2\right)\left(\left\lceil\frac{n}{\alpha}\right\rceil - 3\right).\end{aligned}$$

Notice that

$$\frac{1}{2}\Big(\sqrt{1+2n}-2\Big) \le \alpha \le \frac{1}{2}\Big(\sqrt{1+2n}+2\Big)$$

and hence, $\alpha = \frac{\sqrt{1+2n}}{2} + \epsilon_1$, where $|\epsilon_1| \le 1$. Similarly,

$$\begin{aligned}\frac{2n}{\sqrt{1+2n}+2} &\le \left\lceil \frac{2n}{\sqrt{1+2n}+2} \right\rceil \le \left\lceil \frac{n}{\alpha} \right\rceil \\ &\le \left\lceil \frac{2n}{\sqrt{1+2n}-2} \right\rceil \le \frac{2n}{\sqrt{1+2n}-2} + 1.\end{aligned}$$

which implies that

$$\left\lceil \frac{n}{\alpha} \right\rceil = \frac{2n}{\sqrt{1+2n}} + \epsilon_2,$$

where by simple calculation we can find that $|\epsilon_2| \le 3$. Thus,

$$\begin{aligned}\max_{x\in\mathbb{Z}_2^n} |\mathcal{L}_1(x)| &= (n+1-\alpha)(n-1)+2 - \frac{k}{2}\Big(\left\lceil \frac{n}{\alpha} \right\rceil - 1\Big)\Big(\left\lceil \frac{n}{\alpha} \right\rceil - 2\Big) \\ &\quad - \frac{\alpha-k}{2}\Big(\left\lceil \frac{n}{\alpha} \right\rceil - 2\Big)\Big(\left\lceil \frac{n}{\alpha} \right\rceil - 3\Big) \\ &= (n+1-\alpha)(n-1)+2 - \frac{\alpha}{2}\Big(\left\lceil \frac{n}{\alpha} \right\rceil - 2\Big)\Big(\left\lceil \frac{n}{\alpha} \right\rceil - 3\Big) \\ &\quad - \frac{k}{2}\Big(\left\lceil \frac{n}{\alpha} \right\rceil - 2\Big)\Big(\left\lceil \frac{n}{\alpha} \right\rceil - 1 - \left\lceil \frac{n}{\alpha} \right\rceil + 3\Big) \\ &= (n+1-\alpha)(n-1)+2 - k\Big(\left\lceil \frac{n}{\alpha} \right\rceil - 2\Big) \\ &\quad - \frac{\alpha}{2}\Big(\left\lceil \frac{n}{\alpha} \right\rceil - 2\Big)\Big(\left\lceil \frac{n}{\alpha} \right\rceil - 3\Big) \\ &= (n+1-\frac{\sqrt{1+2n}}{2} - \epsilon_1)(n-1)+2 \\ &\quad - k\Big(\frac{2n}{\sqrt{1+2n}} + \epsilon_2 - 2\Big) \\ &\quad - \frac{\sqrt{1+2n}+2\epsilon_1}{4}\Big(\frac{2n}{\sqrt{1+2n}}+\epsilon_2-2\Big)\Big(\frac{2n}{\sqrt{1+2n}}+\epsilon_2-3\Big) \\ &= n^2 + 1 - \left(\frac{\sqrt{1+2n}}{2} + \epsilon_1\right)(n-1) \\ &\quad - \Big(\frac{2n}{\sqrt{1+2n}}+\epsilon_2-2\Big)\Big(k+\frac{\sqrt{1+2n}+2\epsilon_1}{4}\Big(\frac{2n}{\sqrt{1+2n}}+\epsilon_2-3\Big)\Big).\end{aligned}$$

Note that $1 \le k \le \alpha \le \frac{1}{2}\big(\sqrt{1+2n}+2\big)$, which implies that

$$\begin{aligned}\max_{x\in\mathbb{Z}_2^n} |\mathcal{L}_1(x)| &= n^2 - \frac{n\sqrt{1+2n}}{2} - \frac{n^2}{\sqrt{1+2n}} + O(n) \\ &= n^2 - \sqrt{2}n^{\frac{3}{2}} + O(n).\end{aligned}$$

□

3.5 The Expected Size of an FLL 1-Ball

Let n and $q > 1$ be integers and let $\boldsymbol{x} \in \mathbb{Z}_q^n$. By (3.5), for every $\boldsymbol{x} \in \mathbb{Z}_q^n$, we have

$$\begin{aligned}|\mathcal{L}_1(\boldsymbol{x})| &= \rho(\boldsymbol{x})(n(q-1)-1)+2-\sum_{i=1}^{A(\boldsymbol{x})}\frac{(s_i-1)(s_i-2)}{2}\\ &= \rho(\boldsymbol{x})(nq-n-1)+2-\frac{1}{2}\sum_{i=1}^{A(\boldsymbol{x})}s_i^2+\frac{3}{2}\sum_{i=1}^{A(\boldsymbol{x})}s_i-A(\boldsymbol{x}).\end{aligned}$$

Thus, the average size of an FLL 1-ball, $\mathbb{E}_{\boldsymbol{x}\in\mathbb{Z}_q^n}[|\mathcal{L}_1(\boldsymbol{x})|]$, is

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}\left[\rho(\boldsymbol{x})(n(q-1)-1)+2-\frac{1}{2}\sum_{i=1}^{A(\boldsymbol{x})}s_i^2+\frac{3}{2}\sum_{i=1}^{A(\boldsymbol{x})}s_i-A(\boldsymbol{x})\right]. \tag{3.10}$$

Based on (3.10) and the linearity of the expectation, in order to derive the expected size of an FLL 1-ball, in the following lemmas and claims we analyze the expected values of $\rho(\boldsymbol{x})$, $A(\boldsymbol{x})$, $\sum_{i=1}^{A(\boldsymbol{x})} s_i$, and $\sum_{i=1}^{A(\boldsymbol{x})} s_i^2$.

Lemma 3.27. *For any two integers $n, q > 1$,*

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}\left[\sum_{i=1}^{A(\boldsymbol{x})}s_i\right] = n+(n-2)\cdot\frac{(q-1)(q-2)}{q^2}.$$

Proof. If $\boldsymbol{x} \in \mathbb{Z}_q^n$, then by the definition of an alternating segment, we have that for each $1 \le i \le n$, x_i is contained in at least one maximal alternating segment and not more than two maximal alternating segments. Hence,

$$\sum_{i=1}^{A(\boldsymbol{x})}s_i = n+\zeta(\boldsymbol{x}), \tag{3.11}$$

where $\zeta(\boldsymbol{x})$ denotes the number of entries in $\boldsymbol{x}$ which are contained in exactly two alternating segments. Define, for each $1 \le i \le n$

$$\zeta_i(\boldsymbol{x}) \triangleq \begin{cases}1 & x_i \text{ is contained in two maximal alternating segments}\\ 0 & \text{otherwise}\end{cases} \tag{3.12}$$

Thus,

$$\begin{aligned}\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}\left[\sum_{i=1}^{A(\boldsymbol{x})}s_i\right] &= n+\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}[\zeta(\boldsymbol{x})] = n+\frac{1}{q^n}\sum_{\boldsymbol{x}\in\mathbb{Z}_q^n}\zeta(\boldsymbol{x})\\ &= n+\frac{1}{q^n}\sum_{\boldsymbol{x}\in\mathbb{Z}_q^n}\sum_{i=1}^{n}\zeta_i(\boldsymbol{x})\\ &= n+\frac{1}{q^n}\sum_{i=1}^{n}\sum_{x\in\mathbb{Z}_q^n}\zeta_i(\boldsymbol{x}).\end{aligned}$$

Clearly, if $i \in \{1, n\}$ then $\zeta_i(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in \mathbb{Z}_q^n$. Otherwise, $\zeta_i(\boldsymbol{x}) = 1$ if and only if x_{i-1}, x_i and x_{i+1} are all different. Therefore, for $2 \leq i \leq n-1$, there are $\binom{q}{3} \cdot 3!$ distinct ways to select values for x_{i-1}, x_i, and x_{i+1} and q^{n-3} distinct ways to select values for the other entries of $\boldsymbol{x}$. That is,

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{x} \in \mathbb{Z}_q^n} \left[\sum_{i=1}^{A(\boldsymbol{x})} s_i \right] &= n + \frac{1}{q^n} \sum_{i=1}^{n} \sum_{\boldsymbol{x} \in \mathbb{Z}_q^n} \zeta_i(\boldsymbol{x}) \\
&= n + \frac{1}{q^n} \sum_{i=2}^{n-1} \binom{q}{3} 3! q^{n-3} \\
&= n + (n-2) \cdot \frac{(q-1)(q-2)}{q^2}.
\end{aligned}$$

□

Corollary 3.28. *For $q = 2$, we have that*

$$\mathop{\mathbb{E}}_{\boldsymbol{x} \in \mathbb{Z}_2^n} \left[\sum_{i=1}^{A(\boldsymbol{x})} s_i \right] = n.$$

Before we continue with the calculation of the other expected values, we define the difference vector, which will be used in the analysis to follow.

Definition 3.29. *For a sequence $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$, denote by $\boldsymbol{x}' \in \mathbb{Z}_q^{n-1}$ the* difference vector *of $\boldsymbol{x}$, which is defined by*

$$\boldsymbol{x}' \triangleq (x_2 - x_1, x_3 - x_2, \ldots, x_n - x_{n-1}).$$

Next, we study the expected size of $A(\boldsymbol{x})$. We start by presenting a relation between $A(\boldsymbol{x})$, $\sum_{i=1}^{A(\boldsymbol{x})} s_i$, and the difference vector of $\boldsymbol{x}$ in the next claim.

Claim 3.30. *For integers n and $q > 1$ and a sequence $\boldsymbol{x} \in \mathbb{Z}_q^n$,*

$$\sum_{i=1}^{A(\boldsymbol{x})} s_i = n + A(\boldsymbol{x}) - 1 - \mathsf{Zeros}(\boldsymbol{x}'),$$

where $\mathsf{Zeros}(\boldsymbol{y})$ denotes the number of zeros in $\boldsymbol{y}$.

Proof. By (3.11) we have that

$$\sum_{i=1}^{A(\boldsymbol{x})} s_i = n + \zeta(\boldsymbol{x}).$$

Since there are $A(\boldsymbol{x})$ alternating segments, it follows that there are $A(\boldsymbol{x})$ entries that start with a maximal alternating segment. Denote this set of entries by $\mathsf{Ind}(\boldsymbol{x})$ and let $\mathsf{Ind}_1(\boldsymbol{x}) \subseteq \mathsf{Ind}(\boldsymbol{x})$

be the set of entries $i \in \mathsf{Ind}(\boldsymbol{x})$ that are contained in exactly one maximal alternating segment. This implies that

$$\sum_{i=1}^{A(\boldsymbol{x})} s_i = n + |\mathsf{Ind}(\boldsymbol{x})| - |\mathsf{Ind}_1(\boldsymbol{x})|.$$

Clearly, $1 \in \mathsf{Ind}_1(\boldsymbol{x})$. For any other index $i \in \mathsf{Ind}(\boldsymbol{x})$, x_i is contained in exactly one maximal alternating segment if and only if $x_i = x_{i-1}$, i.e., $x'_{i-1} = 0$. Thus,

$$\sum_{i=1}^{A(\boldsymbol{x})} s_i = n + A(\boldsymbol{x}) - 1 - \mathsf{Zeros}(\boldsymbol{x}').$$

□

Using the latter relation, Lemma 3.27 and the linearity of expectation, the expected value of $A(\boldsymbol{x})$ can be derived from the expected value of $\mathsf{Zeros}(\boldsymbol{x}')$, which is given in the next claim.

Claim 3.31. *Given two integers n and $q > 1$, we have that*

$$\mathop{\mathbb{E}}_{\boldsymbol{x} \in \mathbb{Z}_q^n} \left[\mathsf{Zeros}(\boldsymbol{x}') \right] = \frac{n-1}{q}.$$

Proof. By the definition of the difference vector, given $\boldsymbol{y} \in \mathbb{Z}_q^{n-1}$, the sequence $\boldsymbol{x} \in \Sigma_q^n$ such that $\boldsymbol{x}' = \boldsymbol{y}$ is defined uniquely by the selection of the first entry of $\boldsymbol{x}$ from $\mathbb{Z}_q$. Hence, we have that for each $\boldsymbol{y} \in \mathbb{Z}_q^{n-1}$ there are exactly q sequences $\boldsymbol{x} \in \mathbb{Z}_q^n$ such that $\boldsymbol{x}' = \boldsymbol{y}$. In other words, the function $f(\boldsymbol{x}) = \boldsymbol{x}'$ is a q to 1 function. Define,

$$\mathsf{zero}_i(\boldsymbol{y}) \triangleq \begin{cases} 1 & \text{if } y_i = 0 \\ 0 & \text{otherwise.} \end{cases}$$

It follows that,

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{x} \in \mathbb{Z}_q^n} \left[\mathsf{Zeros}(\boldsymbol{x}') \right] &= \mathop{\mathbb{E}}_{\boldsymbol{y} \in \mathbb{Z}_q^{n-1}} [\mathsf{Zeros}(\boldsymbol{y})] \\
&= \frac{1}{q^{n-1}} \sum_{\boldsymbol{y} \in \mathbb{Z}_q^{n-1}} \mathsf{Zeros}(\boldsymbol{y}) \\
&= \frac{1}{q^{n-1}} \sum_{\boldsymbol{y} \in \mathbb{Z}_q^{n-1}} \sum_{i=1}^{n-1} \mathsf{zero}_i(\boldsymbol{y}) \\
&= \frac{1}{q^{n-1}} \sum_{i=1}^{n-1} \sum_{\boldsymbol{y} \in \mathbb{Z}_q^{n-1}} \mathsf{zero}_i(\boldsymbol{y}).
\end{aligned}$$

For each i, the set $\{\boldsymbol{y} \in \mathbb{Z}_q^{n-1} : y_i = 0\}$ is of size $\frac{q^{n-1}}{q} = q^{n-2}$. Thus,

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{x} \in \mathbb{Z}_q^n} \left[\mathsf{Zeros}(\boldsymbol{x}') \right] &= \frac{1}{q^{n-1}} \sum_{i=1}^{n-1} \sum_{\boldsymbol{y} \in \mathbb{Z}_q^{n-1}} \mathsf{zero}_i(\boldsymbol{y}) \\
&= \frac{1}{q^{n-1}} \cdot \sum_{i=1}^{n-1} q^{n-2} = \frac{n-1}{q}.
\end{aligned}$$

□

By combining the results from Lemma 3.27 and Claims 3.30 and 3.31 we infer the following result.

Corollary 3.32. *For two integers n and $q > 1$, the average number of alternating segments of a sequence $\boldsymbol{x} \in \mathbb{Z}_q^n$ is*

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}[A(\boldsymbol{x})] = 1 + \frac{(n-2)(q-1)(q-2)}{q^2} + \frac{n-1}{q},$$

and in particular for $q = 2$

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_2^n}[A(\boldsymbol{x})] = \frac{n+1}{2}.$$

Proof. For each $q > 1$ we have that

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}[A(\boldsymbol{x})] &= \mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}\left[\sum_{i=1}^{A(\boldsymbol{x})} s_i\right] + \mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}\left[\mathsf{Zeros}(\boldsymbol{x}')\right] - n + 1 \\
&\qquad\qquad \text{(by Claim 3.30)} \\
&= n + \frac{(n-2)(q-1)(q-2)}{q^2} + \frac{n-1}{q} - n + 1 \\
&\qquad\qquad \text{(by Lemma 3.27 and Claim 3.31)} \\
&= 1 + \frac{(n-2)(q-1)(q-2)}{q^2} + \frac{n-1}{q}.
\end{aligned}$$

When $q = 2$ the latter implies the claim. □

The expected size of $\rho(\boldsymbol{x})$ is given in the next lemma.

Lemma 3.33. *For any two integers n and $q > 1$, the average number of runs in a sequence $\boldsymbol{x} \in \mathbb{Z}_q^n$ is*

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}[\rho(\boldsymbol{x})] = n - \frac{n-1}{q}.$$

Proof. For a sequence $\boldsymbol{x} \in \mathbb{Z}_q^n$, the number of runs in $\boldsymbol{x}$ is equal to the number of entries which begin a run in $\boldsymbol{x}$. Clearly, x_1 is the beginning of the first run and by the definition of the difference vector, we have that for each i, $2 \leq i \leq n$, x_i starts a run if and only if $x'_{i-1} \neq 0$. Thus,

$$\rho(\boldsymbol{x}) = n - \mathsf{Zeros}(\boldsymbol{x}'),$$

and, by Claim 3.31,

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}[\rho(\boldsymbol{x})] = n - \mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}\left[\mathsf{Zeros}(\boldsymbol{x}')\right] = n - \frac{n-1}{q}.$$

□

Considering (3.10), our current goal is to evaluate $\mathbb{E}_{\boldsymbol{x}\in\mathbb{Z}_q^n}\left[\sum_{i=1}^{A(\boldsymbol{x})} s_i^2\right]$. Denote by $\chi(s)$ the number of maximal alternating segments of length s over all the sequences $\boldsymbol{x}\in\mathbb{Z}_q^n$, i.e.,

$$\chi(s)=\sum_{\boldsymbol{x}\in\mathbb{Z}_q^n}|\{1\leq i\leq A(\boldsymbol{x})\ :\ s_i=s\}|.$$

It holds that

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}\left[\sum_{i=1}^{A(\boldsymbol{x})} s_i^2\right]=\frac{1}{q^n}\sum_{\boldsymbol{x}\in\mathbb{Z}_2^n}\sum_{i=1}^{A(\boldsymbol{x})} s_i^2=\frac{1}{q^n}\sum_{s=1}^{n} s^2\chi(s),$$

and the values of $\chi(s)$ for $1\leq s\leq n$ are given in the following lemmas.

Lemma 3.34. *If n and $q>1$ are two positive integers then*

$$\chi(1)=2q^{n-1}+(n-2)q^{n-2}.$$

Proof. Let us count the number of maximal alternating segments of length one over all the sequences $\boldsymbol{x}\in\mathbb{Z}_q^n$. Consider the following two cases:
Case 1 - If the alternating segment is at x_1, we can choose the symbols of x_1 in q different ways. Since the alternating segment's length is one, i.e., $x_1=x_2$, it follows that the value of x_2 is determined. The symbols at $x_3,\ldots,x_n$ can be selected in q^{n-2} different ways. Therefore, there are q^{n-1} distinct sequences with such an alternating segment. The same arguments hold for an alternating segment at x_n.
Case 2 - If the alternating segment is at index i, where $2\leq i\leq n-1$, it must be that $x_{i-1}=x_i=x_{i+1}$. The symbol at x_i can be selected in q different ways and the symbols of x_{i-1},x_{i+1} are fixed. In addition. we can set the symbols of $\boldsymbol{x}$ at indices $j\notin\{i-1,i,i+1\}$ in q^{n-3} different ways. Therefore, there are q^{n-2} distinct sequences with such an alternating segment.

Thus,

$$\chi(1)=2q^{n-1}+(n-2)q^{n-2}.$$

□

Lemma 3.35. *For any two integers n and $q>1$,*

$$\chi(n)=q(q-1).$$

Proof. Any alternating segment of length n is defined by the first two symbols which must be distinct (the rest of the symbols are determined by the first two symbols). There are $q(q-1)$ different ways to select the first two symbols and hence the claim follows. □

For $2\leq s\leq n-1$ we need to consider whether the alternating segment overlaps with the preceding or the succeeding segment, or not. To this end, we distinguish between the maximal alternating segments of length s as follows

$\chi_1(s)$ - The number of alternating segments that do not overlap with the preceding segment and the succeeding segments.

$\chi_2(s)$ - The number of alternating segments that overlap with the preceding segment and the succeeding segments.

$\chi_3(s)$ - The number of alternating segments that overlap only with the succeeding segment.

$\chi_4(s)$ - The number of alternating segments that overlap only with the preceding segment.

Claim 3.36. *If $n, q > 1$ are integers and $2 \leq s \leq n-1$ then,*

$$1)\ \chi_1(s) = 2(q-1)q^{n-s} + (n-s-1)(q-1)q^{n-s-1}.$$
$$2)\ \chi_2(s) = (n-s-1)(q-1)(q-2)^2 q^{n-s-1}.$$
$$3)\ \chi_3(s) = (q-1)(q-2)q^{n-s} + (q-1)(q-2)(n-s-1)q^{n-s-1}.$$
$$4)\ \chi_4(s) = (q-1)(q-2)q^{n-s} + (q-1)(q-2)(n-s-1)q^{n-s-1}.$$

Proof. 1) To count the number of maximal alternating segments of length s that do not overlap with the preceding segment and the succeeding segment we distinguish two distinct cases.
Case 1 - If the alternating segment is at the beginning of the sequence, then there are $q(q-1)$ distinct ways to select the symbols of the segment. The symbol after the segment is determined (and is equal to the last symbol of the discussed alternating segment) in order to prevent an overlap and the other symbols can be chosen in q^{n-s-1} different ways. Hence, the number of different sequences with such segments is $(q-1)q^{n-s}$. The same arguments hold for an alternating segment at the end of the sequence.
Case 2 - If the alternating segment is not at the edges of the sequence, then there are $n-s-1$ possible positions to start the alternating segment, and $q(q-1)$ ways to choose the two symbols of the alternating segment. The symbol preceding and the symbol succeeding the alternating segment are determined. The other symbols can be chosen in q^{n-s-2} distinct ways and hence the number of different alternating segments is $(n-s-1)(q-1)q^{n-s-1}$.

Thus,

$$\chi_1(s) = 2(q-1)q^{n-s} + (n-s-1)(q-1)q^{n-s-1}.$$

2) A maximal alternating segment that overlaps with the preceding segment and the succeeding segment cannot be at the sequence edges. Hence, there are $n-s-1$ possible positions to start the alternating segment and the symbols of the segment can be chosen in $q(q-1)$ different ways. In order to overlap with the preceding (succeeding, respectively) segment, the symbol before (after, respectively) the segment must be different from the two symbols of the segment. Therefore, there are $(q-2)^2$ options to choose the symbol before and the symbol after the segment. In addition, the rest of the sequence can be chosen in q^{n-s-2} different ways and hence

$$\chi_2(s) = (n-s-1)(q-1)(q-2)^2 q^{n-s-1}.$$

3) Since the alternating segment must intersect with the succeeding segment, it cannot be the last alternating segment, that is, the segment ends at index $j < n$. To count the number of maximal alternating segments of length s that overlap only with the succeeding segment we consider two distinct cases.
Case 1 - If the alternating segment is at the beginning of the sequence then there are $q(q-1)$ different ways to choose the symbols for it and the symbol after the segment must be different from the two symbols of the alternating segment so there are $(q-2)$ options to select it. The other symbols can be chosen in q^{n-s-1} different ways. Hence, the number of different segments is $(q-1)(q-2)q^{n-s}$.
Case 2 - If the alternating segment does not start at the beginning of the sequence, since the segment ends at index $j < n$, it follows that there are $(n-s-1)$ possible locations to start the segment. There are $q(q-1)$ different ways to select the symbols for the alternating segment. The symbol before the alternating segment is determined in order to prevent an overlap with the previous segment and the symbol after the segment must be different from the two symbols of the alternating segment and hence there are $(q-2)$ ways to choose it. The other symbols can be chosen in q^{n-s-2} different ways and hence the number of different segments is $q^{n-s-1}(q-1)(q-2)(n-s-1)$.
Thus,
$$\chi_3(s) = (q-1)(q-2)q^{n-s} + (q-1)(q-2)(n-s-1)q^{n-s-1}.$$

4) Clearly, the number of maximal alternating segments of length s that overlap only with the succeeding segment is equal to the number alternating segments of length s that overlap only with the preceding segment.

□

Lemma 3.37. *In $n, q > 1$ are integers and $2 \leq s \leq n-1$ then*
$$\chi(s) = 2(q-1)^2 q^{n-s} + (n-s-1)(q-1)^3 q^{n-s-1}.$$

Proof. By Claim 3.36,
$$\begin{aligned}
\chi(s) &= \chi_1(s) + \chi_2(s) + \chi_3(s) + \chi_4(s) \\
&= 2(q-1)q^{n-s} + (n-s-1)(q-1)q^{n-s-1} \\
&\quad + (n-s-1)(q-1)(q-2)^2 q^{n-s-1} \\
&\quad + 2(q-1)(q-2)q^{n-s} \\
&\quad + 2(n-s-1)(q-1)(q-2)q^{n-s-1} \\
&= 2(q-1)^2 q^{n-s} \\
&\quad + (n-s-1)(q-1)q^{n-s-1}\Big(1 + (q-2)^2 + 2(q-2)\Big) \\
&= 2(q-1)^2 q^{n-s} \\
&\quad + (n-s-1)(q-1)q^{n-s-1}\Big(q^2 - 2q + 1\Big) \\
&= 2(q-1)^2 q^{n-s} + (n-s-1)(q-1)^3 q^{n-s-1}.
\end{aligned}$$

□

Lemma 3.38. *If $n, q > 1$ are integers then,*

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}\left[\sum_{i=1}^{A(\boldsymbol{x})} s_i^2\right] = \frac{n(4q^2-3q+2)}{q^2} + \frac{6q-4}{q^2} - 4 - \frac{2}{q-1}\left(1-\frac{1}{q^n}\right).$$

Proof. We have that

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}\left[\sum_{i=1}^{A(\boldsymbol{x})} s_i^2\right] = \frac{1}{q^n}\sum_{\boldsymbol{x}\in\mathbb{Z}_q^n}\sum_{i=1}^{A(\boldsymbol{x})} s_i^2 = \frac{1}{q^n}\sum_{s=1}^{n} s^2\chi(s) = \frac{\chi(1)}{q^n} + \frac{n^2\chi(n)}{q^n} + \frac{1}{q^n}\sum_{s=2}^{n-1} s^2\chi(s).$$

Let us first calculate $\sum_{s=2}^{n-1} s^2\chi(s)$. By Lemma 3.37,

$$\begin{aligned}\sum_{s=2}^{n-1} s^2\chi(s) &= \sum_{s=2}^{n-1} s^2\left(2(q-1)^2q^{n-s} + (n-s-1)(q-1)^3q^{n-s-1}\right)\\ &= 2(q-1)^2\sum_{s=2}^{n-1} s^2q^{n-s}\\ &\quad + (q-1)^3\sum_{s=2}^{n-1}(n-s-1)s^2q^{n-s-1}.\end{aligned}$$

It can be verified that

$$\sum_{s=2}^{n-1} s^2\chi(s) = \frac{2q^3 - q^3n^2(q-1)^2 + q^n(2-2q(3+q(2q-3)))}{(q-1)q^2} + \frac{nq^n(q-1)(1+q(4q-3))}{(q-1)q^2}$$

and after rearranging the latter, we obtain that

$$\sum_{s=2}^{n-1} s^2\chi(s) = nq^{n-2}(4q^2-3q+1) - n^2q(q-1) - 2q^{n-2}\cdot\frac{(2q-1)(q^2-q+1)}{(q-1)} + \frac{2}{q-1} + 2.$$

Hence,

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}\left[\sum_{i=1}^{A(\boldsymbol{x})} s_i^2\right] &= \frac{\chi(1)}{q^n} + \frac{n^2\chi(n)}{q^n} + \frac{1}{q^n}\sum_{s=2}^{n-1} s^2\chi(s) \\
&= \frac{2q^{n-1}+(n-2)q^{n-2}}{q^n} + \frac{n^2q(q-1)}{q^n} \\
&\quad + \frac{nq^{n-2}(4q^2-3q+1)}{q^n} - \frac{n^2q(q-1)}{q^n} \\
&\quad - 2q^{n-2}.\frac{(2q-1)(q^2-q+1)}{q^n(q-1)} + \frac{2}{q^n(q-1)} + \frac{2}{q^n} \\
&= \frac{n(4q^2-3q+2)}{q^2} + \frac{2}{q} - \frac{2}{q^2} \\
&\quad - \frac{2(2q-1)(q^2-q+1)}{q^2(q-1)} + \frac{2}{q^n(q-1)} + \frac{2}{q^n} \\
&= \frac{n(4q^2-3q+2)}{q^2} + \frac{6q-4}{q^2} - 4 \\
&\quad - \frac{2}{q-1}\left(1-\frac{1}{q^n}\right) + \frac{2}{q^n}.
\end{aligned}$$

□

Theorem 3.39. *If $n, q > 1$ are integers, then*

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}[|\mathcal{L}_1(\boldsymbol{x})|] = n^2\left(q+\frac{1}{q}-2\right) - \frac{n}{q} - \frac{(q-1)(q-2)}{q^2} \\
+ 3 - \frac{3}{q} + \frac{2}{q^2} + \frac{q^n-q}{q^n(q-1)}.
\end{aligned}$$

Proof. By (3.10) we have that

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}[|\mathcal{L}_1(\boldsymbol{x})|] &= (nq-n-1)\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}[\rho(\boldsymbol{x})] + 2 \\
&\quad - \frac{1}{2}\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}\left[\sum_{i=1}^{A(\boldsymbol{x})} s_i^2\right] + \frac{3}{2}\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}\left[\sum_{i=1}^{A(\boldsymbol{x})} s_i\right] - \mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}[A(\boldsymbol{x})].
\end{aligned}$$

Using Corollary 3.32 and Lemmas 3.27, 3.33, and 3.38 we infer that

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{x}\in\mathbb{Z}_q^n}[|\mathcal{L}_1(\boldsymbol{x})|] &= (nq-n-1)\left(n-\frac{n-1}{q}\right)+2 \\
&\quad -\frac{1}{2}\left(\frac{n(4q^2-3q+2)}{q^2}+\frac{6q-4}{q^2}-4-\frac{2}{q-1}\left(1-\frac{1}{q^n}\right)+\frac{2}{q^n}\right) \\
&\quad +\frac{3}{2}\left(n+(n-2)\cdot\frac{(q-1)(q-2)}{q^2}\right) \\
&\quad -1-\frac{(n-2)(q-1)(q-2)}{q^2}-\frac{n-1}{q} \\
&= n^2\left(q+\frac{1}{q}-2\right)-\frac{n}{q}-\frac{(q-1)(q-2)}{q^2}+3-\frac{3}{q} \\
&\quad +\frac{2}{q^2}+\frac{q^n-q}{q^n(q-1)}.
\end{aligned}$$

□

3.6 Binary Anticodes with Diameter One

Before presenting the analysis of the anticodes under the FLL metric, we state the following lemma, which was proven in [43, Sections 3 and 5] and will be used in some of the proofs in this section.

Lemma 3.40. *If $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_2^n$ are distinct words, then*

$$|\mathcal{D}_1(\boldsymbol{x}) \cap \mathcal{D}_1(\boldsymbol{y})| \leq 2 \text{ and } |\mathcal{I}_1(\boldsymbol{x}) \cap \mathcal{I}_1(\boldsymbol{y})| \leq 2.$$

Definition 3.41. *An* anticode of diameter *t in $\mathbb{Z}_q^n$ is a subset $\mathcal{A} \subseteq \mathbb{Z}_q^n$ such that for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{A}$, $d_\ell(\boldsymbol{x}, \boldsymbol{x}') \leq t$. We say that $\mathcal{A}$ is a* maximal anticode *if there is no other anticode of diameter t in $\mathbb{Z}_q^n$ which contains $\mathcal{A}$.*

Next, we present tight lower and upper bounds on the size of maximal binary anticodes of diameter one in the FLL metric. To prove these bounds we need some useful properties of anticodes with diameter one in the FLL metric.

Lemma 3.42. *If an anticode $\mathcal{A}$ of diameter one contains three distinct words with the suffix 00 then there is at most one word in $\mathcal{A}$ with the suffix 01.*

Proof. Let $\boldsymbol{a}, \boldsymbol{a}', \boldsymbol{a}'' \in \mathcal{A}$ be three words with the suffix 00 and assume to the contrary that there exist two distinct words $\boldsymbol{b}, \boldsymbol{b}' \in \mathcal{A}$ with the suffix 01. Let $\boldsymbol{y} \in \mathcal{LCS}(\boldsymbol{a}, \boldsymbol{b})$; by Corollary 3.8 the length of $\boldsymbol{y}$ is $n-1$ and since $\boldsymbol{a}$ ends with 00, $\boldsymbol{y}$ must end with 0 which implies that $\boldsymbol{y} = \boldsymbol{b}_{[1,n-1]}$. By the same arguments $\boldsymbol{y} \in \mathcal{LCS}(\boldsymbol{b}, \boldsymbol{a}')$ and $\boldsymbol{y} \in \mathcal{LCS}(\boldsymbol{b}, \boldsymbol{a}'')$. Similarly,

$$\boldsymbol{y}' = \boldsymbol{b}'_{[1,n-1]} \in \mathcal{LCS}(\boldsymbol{b}', \boldsymbol{a}, \boldsymbol{a}', \boldsymbol{a}'').$$

Hence, $\boldsymbol{a}, \boldsymbol{a}', \boldsymbol{a}'' \in \mathcal{I}_1(\boldsymbol{y}) \cap \mathcal{I}_1(\boldsymbol{y}')$ which is a contradiction to Lemma 3.40. Thus, $\mathcal{A}$ contains at most one word with the suffix 01. □

Lemma 3.43. *If an anticode $\mathcal{A}$ of diameter one contains three distinct words with the suffix 01, then there is at most one word in $\mathcal{A}$ with the suffix 00.*

Proof. Let $\boldsymbol{a}, \boldsymbol{a}', \boldsymbol{a}'' \in \mathcal{A}$ be three words with the suffix 01 and assume to the contrary that there exist two distinct words $\boldsymbol{b}, \boldsymbol{b}' \in \mathcal{A}$ with the suffix 00. For $\boldsymbol{y} \in \mathcal{LCS}(\boldsymbol{a}, \boldsymbol{b})$, by Corollary 3.8 the length of $\boldsymbol{y}$ is $n-1$ and since $\boldsymbol{b}$ ends with 00, $\boldsymbol{y}$ must end with 0 which implies that $\boldsymbol{y} = \boldsymbol{a}_{[1,n-1]}$. By the same arguments $\boldsymbol{y} \in \mathcal{LCS}(\boldsymbol{a}, \boldsymbol{b}')$. Similarly,

$$\begin{aligned}\boldsymbol{y}' &= \boldsymbol{a}'_{[1,n-1]} \in \mathcal{LCS}(\boldsymbol{a}', \boldsymbol{b}, \boldsymbol{b}') \\ \boldsymbol{y}'' &= \boldsymbol{a}''_{[1,n-1]} \in \mathcal{LCS}(\boldsymbol{a}'', \boldsymbol{b}, \boldsymbol{b}').\end{aligned}$$

Hence, $\boldsymbol{y}, \boldsymbol{y}', \boldsymbol{y}'' \in \mathcal{D}_1(\boldsymbol{b}) \cap \mathcal{D}_1(\boldsymbol{b}')$ which is a contradiction to Lemma 3.40. Thus, $\mathcal{A}$ contains at most one word with the suffix 00. □

Lemma 3.44. *Let $\mathcal{A}$ be an anticode of diameter one. If $\boldsymbol{a}, \boldsymbol{a}' \in \mathcal{A}$ are two distinct words that end with 00 and $\boldsymbol{b}, \boldsymbol{b}' \in \mathcal{A}$ are two distinct words that end with 01, then $\boldsymbol{a}_{[1,n-1]} \neq \boldsymbol{b}_{[1,n-1]}$ or $\boldsymbol{a}'_{[1,n-1]} \neq \boldsymbol{b}'_{[1,n-1]}$.*

Proof. Assume to the contrary that there exist $\boldsymbol{a}, \boldsymbol{a}', \boldsymbol{b}, \boldsymbol{b}' \in \mathcal{A}$ such that $\boldsymbol{a}_{[1,n-1]} = \boldsymbol{b}_{[1,n-1]} = \boldsymbol{y}0$ and $\boldsymbol{a}'_{[1,n-1]} = \boldsymbol{b}'_{[1,n-1]} = \boldsymbol{y}'0$, $\boldsymbol{a}, \boldsymbol{a}'$ end with 00 and $\boldsymbol{b}, \boldsymbol{b}'$ end with 01. Let

$$\begin{aligned}\boldsymbol{a} &= a_1\, a_2 \ldots a_{n-2}\, 0\, 0 = \boldsymbol{y}\ 0\, 0 \\ \boldsymbol{a}' &= a'_1\, a'_2 \ldots a'_{n-2}\, 0\, 0 = \boldsymbol{y}'\, 0\, 0 \\ \boldsymbol{b} &= a_1\, a_2 \ldots a_{n-2}\, 0\, 1 = \boldsymbol{y}\ 0\, 1 \\ \boldsymbol{b}' &= a'_1\, a'_2 \ldots a'_{n-2}\, 0\, 1 = \boldsymbol{y}'\, 0\, 1.\end{aligned}$$

Notice that since the FLL distance between any two words in $\mathcal{A}$ is one, it follows that the Hamming weight of any two words can differ by at most one, which implies that $\mathrm{wt}(\boldsymbol{y}) = \mathrm{wt}(\boldsymbol{y}')$ (by considering the pairs $\boldsymbol{a}, \boldsymbol{b}'$ and $\boldsymbol{a}', \boldsymbol{b}$). Clearly, $\boldsymbol{y}0 \in \mathcal{LCS}(\boldsymbol{a}', \boldsymbol{b})$ which implies that $\boldsymbol{a}'$ can be obtained from $\boldsymbol{b}$ by deleting the last 1 of $\boldsymbol{b}$ and then inserting 0 into the LCS. Hence, there exists an index $0 \leq j \leq n-2$ such that

$$a_1 a_2 \ldots a_j 0 a_{j+1} \ldots a_{n-2} 0 = a'_1 a'_2 \ldots a'_j a'_{j+1} \ldots a'_{n-2} 00. \tag{3.13}$$

Similarly, $\boldsymbol{a}$ can be obtained from $\boldsymbol{b}'$, i.e., there exists an index $0 \leq i \leq n-2$ such that

$$a'_1 a'_2 \ldots a'_i 0 a'_{i+1} \ldots a'_{n-2} 0 = a_1 a_2 \ldots a_i a_{i+1} \ldots a_{n-2} 00. \tag{3.14}$$

Assume w.l.o.g. that $i \leq j$. Equation (3.13) implies that $a_r = a_{r'}$ for $1 \leq r \leq j$. In addition, $a_{n-2} = 0$ by (3.13) and $a'_{n-2} = 0$ by (3.14). By assigning $a_{n-2} = a'_{n-2} = 0$ into (3.13) and (3.14) we obtain that $a_{n-3} = a'_{n-3} = 0$. Repeating this process implies that $a_r = a_{r'} = 0$ for $j+1 \leq r \leq n-2$. Thus, we have that $\boldsymbol{y} = \boldsymbol{y}'$ which is a contradiction. □

Definition 3.45. *For an anticode $\mathcal{A} \subseteq \mathbb{Z}_2^n$, the* puncturing of $\mathcal{A}$ in the n-th coordinate*, $\mathcal{A}'$, is defined by*

$$\mathcal{A}' \triangleq \left\{ \boldsymbol{a}_{[1,n-1]} \ : \ \boldsymbol{a} \in \mathcal{A} \right\}.$$

Lemma 3.46. *Let $\mathcal{A} \subseteq \mathbb{Z}_2^n$ be an anticode of diameter one. If the last symbol in all the words in $\mathcal{A}$ is the same symbol $\sigma \in \mathbb{Z}_2$, then $\mathcal{A}'$ is an anticode of diameter one and $|\mathcal{A}'| = |\mathcal{A}|$.*

Proof. Let $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{A}$ be two different words and let $\boldsymbol{y} \in \mathcal{LCS}(\boldsymbol{a}_{[1,n-1]}, \boldsymbol{b}_{[1,n-1]})$. By (3.6), $\mathsf{LCS}(\boldsymbol{a}, \boldsymbol{b}) \leq |\boldsymbol{y}| + 1$ and since $d_\ell(\boldsymbol{a}, \boldsymbol{b}) = 1$, Corollary 3.8 implies that $|\boldsymbol{y}| \geq n - 2$ and that

$$d_\ell(\boldsymbol{a}_{[1,n-1]}, \boldsymbol{b}_{[1,n-1]}) \leq 1.$$

Hence, $\mathcal{A}$ is an anticode of diameter one. Since any two distinct words $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{A}$ end with the symbol σ, it follows that $\boldsymbol{a}_{[1,n-1]} \neq \boldsymbol{b}_{[1,n-1]}$ and thus $|\mathcal{A}| = |\mathcal{A}'|$. □

Lemma 3.47. *Let $\mathcal{A}$ be an anticode of diameter one. If the suffix of each word in $\mathcal{A}$ is either 01 or 10, then $\mathcal{A}'$ is an anticode of diameter one and $|\mathcal{A}'| = |\mathcal{A}|$.*

Proof. Let $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{A}$ be two different words and let $\boldsymbol{y} \in \mathcal{LCS}(\boldsymbol{a}_{[1,n-1]}, \boldsymbol{b}_{[1,n-1]})$. By (3.6), $\mathsf{LCS}(\boldsymbol{a}, \boldsymbol{b}) \leq |\boldsymbol{y}| + 1$ and since $d_\ell(\boldsymbol{a}, \boldsymbol{b}) = 1$, it follows that $|\boldsymbol{y}| \geq n - 2$ and that

$$d_\ell(\boldsymbol{a}_{[1,n-1]}, \boldsymbol{b}_{[1,n-1]}) \leq 1.$$

Hence, $\mathcal{A}'$ is an anticode of diameter one. If $\boldsymbol{a}$ and $\boldsymbol{b}$ end with the same symbol $\sigma \in \{0, 1\}$, then $\boldsymbol{a}_{[1,n-1]} \neq \boldsymbol{b}_{[1,n-1]}$. Otherwise, one of the words has the suffix 01 and the other has the suffix 10. That is, $a_{n-1} \neq b_{n-1}$ and therefore $\boldsymbol{a}_{[1,n-1]} \neq \boldsymbol{b}_{[1,n-1]}$ and thus, $|\mathcal{A}'| = |\mathcal{A}|$. □

3.6.1 Upper Bound

Theorem 3.48. *Let $n > 1$ be an integer and let $\mathcal{A} \subseteq \mathbb{Z}_2^n$ be a maximal anticode of diameter one. Then, $|\mathcal{A}| \leq n + 1$, and there exists a maximal anticode with exactly $n + 1$ codewords.*

Proof. Since two words $\boldsymbol{x}, \boldsymbol{y}$ such that $\boldsymbol{x}$ ends with 00 and $\boldsymbol{y}$ ends with 11 are at FLL distance at least 2, w.l.o.g. assume that $\mathcal{A}$ does not contain codewords that end with 11. It is easy to verify that the theorem holds for $n \in \{2, 3, 4\}$. Assume that the theorem does not hold and let $n^* > 4$ be the smallest integer such that there exists an anticode $\mathcal{A} \subseteq \mathbb{Z}_2^{n^*}$ such that $|\mathcal{A}| = n^* + 2$. Since there are only three possible options for the last two symbols of codewords in $\mathcal{A}$ (00, 01, or 10) and $|\mathcal{A}| \geq 7$, it follows that there exist three different codewords in $\mathcal{A}$ with the same suffix of two symbols.

Case 1 - Assume $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathcal{A}$ are three different words with the suffix 00. By Lemma 3.42, there exists at most one codeword in $\mathcal{A}$ with the suffix 01 and since $\mathcal{A}$ does not contain codewords with the suffix 11, there exists at most one codeword in $\mathcal{A}$ that ends with the symbol 1. That is, there exist at least $n^* + 1$ codewords with 0 as the last symbol. Denote such a set with $n^* + 1$ codewords by $\mathcal{A}_1$. As a subset of the anticode $\mathcal{A}$, $\mathcal{A}_1$ is also an anticode and hence by Lemma 3.46, $\mathcal{A}_1'$ is an anticode of length $n^* - 1$ and size $n^* + 1$ which is a contradiction to the minimality of n^*.

Case 2 - Assume $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathcal{A}$ are three different words with the suffix 01. By Lemma 3.43, there exists at most one codeword in $\mathcal{A}$ with the suffix 00 and since $\mathcal{A}$ does not contain codewords with the suffix 11 there exist $n^* + 1$ codewords that end with either 01 or 10. Denote this set of $n^* + 1$ codewords as $\mathcal{A}_1$. As a subset of the anticode $\mathcal{A}$, $\mathcal{A}_1$ is also an anticode and hence by Lemma 3.47, $\mathcal{A}_1'$ is an anticode of length $n^* - 1$ and size $n^* + 1$ which is a

contradiction to the minimality of n^*.

Case 3 - Assume $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathcal{A}$ are three different words with the suffix 10. By the previous two cases, there exist at most two codewords in $\mathcal{A}$ with the suffix 00 and at most two codewords with the suffix 01. Since there are no codewords with the suffix 11, it follows that the number of words that end with 1 is at most two. If there exist at most one codeword in $\mathcal{A}$ that ends with 1, then there are $n^* + 1$ codewords in $\mathcal{A}$ that end with 0 and as in the first case, this leads to a contradiction. Otherwise there are exactly two codewords in $\mathcal{A}$ with the suffix 01. If there are less than two codewords with the suffix 00, then, the number of codewords with suffixes 01 and 10 is at least $n^* + 1$ and similarly to Case 2, this is a contradiction to the minimality of n^*. Hence, there exist exactly two codewords in $\mathcal{A}$ with the suffix 00. There are exactly $n^* - 2$ codewords in $\mathcal{A}$ with the suffix 10 and two more codewords with the suffix 01. By Lemma 3.47 the words in $\mathcal{A}'$ that were obtained from these n^* codewords are all different and have FLL distance one from each other. In addition, by Lemma 3.44, the prefix of length $n^* - 1$ of at least one of the codewords that end with 00 is different from the prefixes of length $n^* - 1$ of the codewords that end with 01. This prefix also differs from the prefixes of the codewords that end with 10. Therefore, $\mathcal{A}'$ is an anticode with $n^* + 1$ different codewords which is a contradiction to the minimality of n^*.

Note that the set $\mathcal{A} = \{a \in \mathbb{Z}_2^n : \mathrm{wt}(a) \leq 1\}$ is an anticode of diameter one with exactly $n + 1$ codewords. Thus, the maximum size of an anticode of diameter one is $n + 1$. □

3.6.2 Lower Bound

Theorem 3.49. *Let $n > 2$ be a positive integer and let $\mathcal{A} \subseteq \mathbb{Z}_2^n$ be a maximal anticode of diameter one, then $|\mathcal{A}| \geq 4$ and there exists a maximal anticode with exactly 4 codewords.*

Proof. For $n = 3$ the maximal anticodes are

$$\mathcal{A}_1 = \{000,\ 001,\ 010,\ 100\}\ \mathcal{A}_2 = \{001,\ 010,\ 100,\ 101\}$$
$$\mathcal{A}_3 = \{001,\ 010,\ 011,\ 101\}\ \mathcal{A}_4 = \{010,\ 011,\ 101,\ 110\}$$
$$\mathcal{A}_5 = \{011,\ 101,\ 110,\ 111\}\ \mathcal{A}_6 = \{010,\ 100,\ 101,\ 110\}$$

and all of them have size $4 = n + 1$. Assume that the theorem does not hold and let $n^* > 3$ be the smallest integer such that there exists a maximal anticode $\mathcal{A} \subseteq \mathbb{Z}_2^{n^*}$ with less than four codewords. For each $\boldsymbol{x} \in \mathbb{Z}_2^{n^*}$ there exists a sequence $\boldsymbol{y} \in \mathbb{Z}_2^{n^*}$ such that $d_\ell(\boldsymbol{x}, \boldsymbol{y}) = 1$ and hence $|\mathcal{A}| > 1$. If $\mathcal{A} = \{\boldsymbol{x}, \boldsymbol{y}\} \subseteq \mathbb{Z}_2^{n^*}$ by the definition of an anticode $d_\ell(\boldsymbol{x}, \boldsymbol{y}) = 1$ and $\mathsf{LCS}(\boldsymbol{x}, \boldsymbol{y}) = n - 1$. For $\boldsymbol{z} \in \mathcal{LCS}(\boldsymbol{x}, \boldsymbol{y})$, by (3.2), the insertion ball of radius one centered at $\boldsymbol{z}$ contains $n^* - 1 > 2$ codewords in addition to $\boldsymbol{x}$ and $\boldsymbol{y}$ and each of them can be added into $\mathcal{A}$. Hence, $\mathcal{A}$ is an anticode of diameter one with three codewords. We will prove that there exists a word that can be added into $\mathcal{A}$ which is a contradiction to the maximality of $\mathcal{A}$. Consider the following cases:

Case 1 - If all the codewords in $\mathcal{A}$ have the same last symbol $\sigma \in \mathbb{Z}_2$, then by Lemma 3.46, $\mathcal{A}' \subseteq \mathbb{Z}_2^{n^*-1}$, is an anticode of diameter one that contains three codewords. Since n^* is the smallest integer for which there exists a maximal anticode with less than four codewords, $\mathcal{A}'$ is not maximal. That is, there exists a word $\boldsymbol{x}' \in \mathbb{Z}_2^{n^*-1}$ such that $\mathcal{A}' \cup \{\boldsymbol{x}'\}$ is an anticode of diameter one. It can be readily verified that $\boldsymbol{x}'\sigma \notin \mathcal{A}$ and that $\mathcal{A} \cup \{\boldsymbol{x}'\sigma\}$ is an anticode of

diameter one which is a contradiction to the maximality of $\mathcal{A}$.
Case 2 - If all the codewords in $\mathcal{A}$ have the same first symbol $\sigma \in \mathbb{Z}_2$ then a contradiction is obtained by symmetrical arguments to those presented in Case 1.
Case 3 - Assume all the words in $\mathcal{A}$ neither have the same first symbol nor the same last symbol. Let $|\mathcal{A}| = \{x, y, z\}$ and assume w.l.o.g. that x and y are codewords that end with 0 and that z ends with 1. If $|\mathcal{A}'| = 3$, then $z_{[1,n^*-1]} \neq x_{[1,n^*-1]}$ and $z_{[1,n^*-1]} \neq y_{[1,n^*-1]}$. Hence the word $z_{[1,n^*-1]}0$ is not in $\mathcal{A}$ and it is easy to verify that it has distance one from each codeword in $\mathcal{A}$, which is a contradiction. Otherwise, since $x_{[1,n^*-1]} \neq y_{[1,n^*-1]}$, it must hold that $z_{[1,n^*-1]}$ is equal either to $x_{[1,n^*-1]}$ or to $y_{[1,n^*-1]}$. Assume w.l.o.g. that $z_{[1,n^*-1]} = x_{[1,n^*-1]}$, then x and z have the same first symbol σ and hence y must begin with $\overline{\sigma} = 1 - \sigma$. The three codewords can be described as follows:

$$\begin{aligned} x &= \sigma x_2 x_3 \dots x_{n^*-1} 0 \\ y &= \overline{\sigma} y_2 y_3 \dots y_{n^*-1} 0 \\ z &= \sigma x_2 x_3 \dots x_{n^*-1} 1. \end{aligned}$$

Since y and z have different first and last symbols, their LCS must be equal to the suffix of length $n^* - 1$ of one word and to the prefix of length $n^* - 1$ of the other word. If

$$z_{[1,n^*-1]} = \sigma a_2 a_3 \dots a_{n^*-1} \in \mathcal{LCS}(y, z),$$

then $z_{[1,n^*-1]}$ is a common LCS of the three codewords x, y, and z and hence any word from $\mathcal{I}_1(z_{[1,n^*-1]})$ has distance one from all the words in $\mathcal{A}$. Since, by (3.2),

$$|\mathcal{I}_1(z_{[1,n^*-1]})| = n^* + 1 \geq 4,$$

there is a word different from x, y and z that can be added into $\mathcal{A}$. In the other case,

$$\begin{aligned} \overline{\sigma} y_2 y_3 \dots y_{n^*-1} &= y_{[1,n-1]} = z_{[2,n]} \\ &= x_2 x_3 \dots x_{n^*-1} 1 \in \mathcal{LCS}(y, z) \end{aligned}$$

and hence the codewords x and z can be written as

$$\begin{aligned} x &= \sigma \overline{\sigma} y_2 y_3 \dots y_{n^*-2} 0 \\ z &= \sigma \overline{\sigma} y_2 y_3 \dots y_{n^*-2} y_{n^*-1} \end{aligned}$$

and the word

$$w = \sigma \overline{\sigma} y_2 y_3 \dots y_{n^*-1} 0$$

is a common SCS of x, y, and z. If $\rho(w) > 3$ then there is a word in $\mathcal{D}_1(w)$ that is different from x, y, and z that can be added into $\mathcal{A}$, which is again a contradiction. Otherwise, since the first two symbols of w are different and the last two symbols are also different, it holds that $\rho(w) = 3$. It is easy to verify that

$$\mathcal{A} = \{0 \underbrace{11\dots1}_{n^*-2 \text{ times}} 0, 0 \underbrace{11\dots1}_{n^*-1 \text{ times}}, \underbrace{11\dots1}_{n^*-1 \text{ times}} 0\}$$

and that $\underbrace{11\ldots1}_{n^*-2 \text{ times}}01$ can be added into $\mathcal{A}$, which is a contradiction to the minimality of $\mathcal{A}$.
To see that the given bound is tight, one can simply consider the set of codewords that consist from the binary representation of length n^* of the numbers $2, 3, 5, 6$ that is, the set

$$\mathcal{A} = \{\underbrace{0\ldots0}_{n^*-3}010,\ \underbrace{0\ldots0}_{n^*-3}011,\ \underbrace{0\ldots0}_{n^*-3}101,\ \underbrace{0\ldots0}_{n^*-3}110\}$$

and verify that it is indeed a maximal anticode of diameter one. □

3.7 Conclusion

In this paper we studied the size of balls with radius one and the anticodes of diameter one under the FLL metric. In particular we give explicit expressions for the maximum size of a ball with radius one and the minimum size of a ball of any given radius in the FLL metric over $\mathbb{Z}_q$. We also found the average size of a 1-ball in the FLL metric. Finally, we considered the related concept of anticode in the FLL distance and we found that the maximum and minimum size of a binary maximal anticode of diameter one are $n+1$ and 4, respectively. The latter can be extended to a non-binary alphabet and while the minimum size of a maximal anticode with diameter one is 4 for any alphabet size q, the maximum size of a maximal anticode with diameter one is $n(q-1)+1$. Recently, based on these results, G. Wang and Q. Wang [36] extended the analysis of 1-FLL balls by proving that the size of the 1-FLL balls is highly concentrated around its mean using Azuma's inequality [1]. A future direction is to study the maximum size of FLL balls for larger radii, and in particular for radius two. Based on a computer search, it appears that the maximum balls are again centered at α-balanced sequences, however, understanding the value of α for this case is more challenging. For example,

- for $n = 16$, the maximum ball is centered at

  $$\boldsymbol{x} = 0101101001011010,$$

  which is a 4-balanced sequence for which $|\mathcal{L}_2(\boldsymbol{x})| = 4,513$.

- for $n = 20$, the maximum ball is centered at

  $$\boldsymbol{x} = 01010010100101001010,$$

  which is a 4-balanced sequence for which $|\mathcal{L}_2(\boldsymbol{x})| = 12,759$.

- for $n = 25$, the maximum ball is centered at

  $$\boldsymbol{x} = 0101001010010100101001010,$$

  which is a 5-balanced sequence for which $|\mathcal{L}_2(\boldsymbol{x})| = 35,893$.

Acknowledgement

The authors would like to thank the associate editor Xiande Zhang and the two anonymous reviewers for their very helpful comments.

Bibliography

[1] N. Alon and J. H. Spencer, *The Probabilistic Method,* John Wiley & Sons, 2016.

[2] L. Anavy, I. Vaknin, O.Atar, R. Amit and Z. Yakhini, "Data storage in DNA with fewer synthesis cycles using composite DNA letters". *Nature Biotechnology,* vol. 37, no. 10, pp. 1229–1236, 2019.

[3] D. Bar-Lev, T. Etzion, and E. Yaakobi, "On Levenshtein balls with radius one," *2021 IEEE International Symposium Information Theory*, Jul. 2021, pp. 1979–1984.

[4] D. Bar-Lev, I. Orr, O. Sabary, T. Etzion, and E. Yaakobi, "Deep DNA storage: Scalable and robust DNA storage via coding theory and deep learning," *arXiv preprint arXiv:2109.00031*, Sep. 2021.

[5] J. Brakensiek, V. Guruswami, and S. Zbarsky, "Efficient low-redundancy codes for correcting multiple deletions," *Annual ACM-SIAM Symposium on Discrete Algorithms*, 2016, pp. 1884–1892.

[6] B. Bukh, V. Guruswami and J. Håstad, "An improved bound on the fraction of correctable deletions," *IEEE Transactions on Information Theory,* vol. 63, no. 1, pp. 93–103, Jan. 2017.

[7] J. Castiglione and A. Kavcic, "Trellis based lower bounds on capacities of channels with synchronization errors," *2015 IEEE Information Theory Workshop (ITW)*, May 2015, pp. 24–28.

[8] M. Cheraghchi, "Capacity upper bounds for deletion-type channels," *Journal of the ACM (JACM)*, vol. 66, no.2, pp. 1–79, 2019.

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

[10] R. Con and A. Shpilka, "Explicit and efficient constructions of coding schemes for the binary deletion channel and the poisson repeat channel," *2020 IEEE International Symposium Information Theory (ISIT)*, Jun. 2020, pp. 84–89.

[11] D. Cullina and N. Kiyavash, "An improvement to Levenshtein's upper bound on the cardinality of deletion correcting codes," *2013 IEEE International Symposium Information Theory (ISIT)*, 2013, pp. 699–703.

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

[13] R. Gabrys and F. Sala, "Codes correcting two deletions," *IEEE Transactions on Information Theory*, vol. 65, no. 2, pp. 965–974, Feb. 2019.

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

[15] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angewandte Chemie International Edition* vol. 54, no. 8, pp. 2552–2555, Feb. 2015.

[16] V. Guruswami and C. Wang, "Deletion codes in the high-noise and high-rate regimes," *IEEE Transactions on Information Theory*, vol. 63, no. 4, pp. 1961–1970, Apr. 2017.

[17] R. Heckel, G. Mikutis, and R.N. Grass, "A characterization of the DNA data storage channel," *Scientific Reports*, vol. 9, no. 1, p. 9663, 2019.

[18] R. Heckel, I. Shomorony, K. Ramchandran and D. N. C. Tse, "Fundamental limits of DNA storage systems," *2017 IEEE International Symposium Information Theory (ISIT)*, 2017, pp. 3130–3134.

[19] D. S. Hirschberg and M. Regnier, "Tight bounds on the number of string subsequences," *Journal of Discrete Algorithms*, vol. 1, no. 1, pp. 123–132, 2000.

[20] S. Y. Itoga, "The string merging problem," *BIT Numerical Mathematics*, vol. 21, no. 1, pp. 20–30, 1981.

[21] A. Lenz, P. H. Siegel, A. Wachter-Zeh and E. Yaakobi, "On the capacity of DNA-based data storage under substitution errors," *International Conference on Visual Communications and Image Processing (VCIP)*, 2021, pp. 1–5.

[22] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics. Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[23] V. I. Levenshtein, "Efficient reconstruction of sequences from their subsequences or supersequences," *Journal of Combinatorial Theory. Series A*, vol. 93, no. 2, pp. 310–332, 2001.

[24] Y. Liron and M. Langberg, "A characterization of the number of subsequences obtained via the deletion channel," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2300–2312, May 2015.

[25] M. Mitzenmacher, "A survey of results for deletion channels and related synchronization channels," *Probability Surveys* vol. 6, pp. 1–33, 2009.

[26] M. Mitzenmacher and E. Drinea, "A simple lower bound for the capacity of the deletion channel," *IEEE Transactions on Information Theory,* vol. 52, no. 10, pp. 4657–4660, Oct. 2006.

[27] L. Organick et al., "Random access in large-scale DNA data storage," *Nature Biotechnology*, vol. 36, no. 3, pp. 242–248, 2018.

[28] M. Rahmati and T. M. Duman, "Upper bounds on the capacity of deletion channels using channel fragmentation," *IEEE Transactions on Information Theory*, vol. 61, no. 1, pp. 146–156, Jan. 2015.

[29] O. Sabary, Y. Orlev, R. Shafir, L. Anavy, E. Yaakobi, and Z. Yakhini "SOLQC: Synthetic oligo library quality control tool," *Bioinformatics*, vol. 37, no. 5, pp. 720–722, 2021.

[30] F. Sala and L. Dolecek, "Counting sequences obtained from the synchronization channel," *2013 IEEE International Symposium on Information Theory (ISIT)*, 2013, pp. 2925–2929.

[31] F. Sala, R. Gabrys, and L. Dolecek, "Gilbert-Varshamov-like lower bounds for deletion-correcting codes," *2014 IEEE Information Theory Workshop (ITW)*, 2014, pp. 147–151.

[32] J. Sima and J. Bruck, "Optimal k-deletion correcting codes," *2019 IEEE International Symposium Information Theory (ISIT)*, 2019, pp. 847–851.

[33] J. Sima, N. Raviv, and J. Bruck, "On coding over sliced information," *IEEE Transactions on Information Theory*, vol. 67, no. 5, pp. 2793–2807, 2021.

[34] S. K. Tabatabaei et al., "DNA punch cards for storing data on native DNA sequences via enzymatic nicking," *Nature communications*, vol. 11, no. 1, p. 1742, 2020.

[35] I. Tal, H. D. Pfister, A. Fazeli, and A. Vardy, "Polar codes for the deletion channel: Weak and strong polarization," *2019 IEEE International Symposium Information Theory (ISIT)*, 2019, pp.1362–1366.

[36] G. Wang and Q. Wang, "On the size distribution of the fixed-length Levenshtein balls with radius one," *Designs, Codes and Cryptography*, pp. 1–13, 2024.

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

Appendix

Proof of Lemma 3.24

Let $\alpha > 1$ be some integer and define $\text{diff} \triangleq \left|L_1\left(\boldsymbol{x}^{(\alpha)}\right)\right| - \left|L_1\left(\boldsymbol{x}^{(\alpha-1)}\right)\right|$. We will prove that $\text{diff} > 0$ if and only if $n > 2\alpha(\alpha-1)$ by proving that $\text{diff} > 0$ for any $n > 2\alpha(\alpha-1)$ and that $\text{diff} < 0$ for any $\alpha < n < 2\alpha(\alpha-1)$. Before we analyze each case we present different expressions for $\left|L_1\left(\boldsymbol{x}^{(\alpha)}\right)\right|$ that will be in use within the proof. By Lemma 3.23, if n is divisible by α, then

$$\begin{aligned}\left|\mathcal{L}_1\left(\boldsymbol{x}^{(\alpha)}\right)\right| &= (n+1-\alpha)(n-1) + 2 - \frac{\alpha}{2}\left(\frac{n}{\alpha}-1\right)\left(\frac{n}{\alpha}-2\right) \\ &= (n+1-\alpha)(n-1) + 2 - \frac{n^2}{2\alpha} + \frac{3n}{2} - \alpha.\end{aligned}$$

Otherwise when n is not divisible by α, we have that $\left\lceil \frac{n}{\alpha} \right\rceil = \frac{n-k_\alpha}{\alpha} + 1$ and hence, by Lemma 3.23,

$$\begin{aligned}\left|\mathcal{L}_1\left(\boldsymbol{x}^{(\alpha)}\right)\right| &= (n+1-\alpha)(n-1)+2 - \frac{k_\alpha}{2}\left(\left\lceil\frac{n}{\alpha}\right\rceil - 1\right)\left(\left\lceil\frac{n}{\alpha}\right\rceil - 2\right) \\ &\quad - \frac{\alpha-k_\alpha}{2}\left(\left\lceil\frac{n}{\alpha}\right\rceil - 2\right)\left(\left\lceil\frac{n}{\alpha}\right\rceil - 3\right) \\ &= (n+1-\alpha)(n-1)+2 - \frac{k_\alpha}{2}\left(\frac{n-k_\alpha}{\alpha}\right)\left(\frac{n-k_\alpha}{\alpha}-1\right) \\ &\quad - \frac{\alpha-k_\alpha}{2}\left(\frac{n-k_\alpha}{\alpha}-1\right)\left(\frac{n-k_\alpha}{\alpha}-2\right) \\ &= (n+1-\alpha)(n-1)+2 \\ &\quad - \frac{k_\alpha}{2}\left(\frac{n-k_\alpha}{\alpha}-1\right)\left(\frac{n-k_\alpha}{\alpha} - \frac{n-k_\alpha}{\alpha}+2\right) \\ &\quad - \frac{\alpha}{2}\left(\frac{n-k_\alpha}{\alpha}-1\right)\left(\frac{n-k_\alpha}{\alpha}-2\right) \\ &= (n+1-\alpha)(n-1)+2 - k_\alpha\left(\frac{n-k_\alpha}{\alpha}-1\right) \\ &\quad - \frac{\alpha}{2}\left(\frac{n-k_\alpha}{\alpha}-1\right)\left(\frac{n-k_\alpha}{\alpha}-2\right) \\ &= (n+1-\alpha)(n-1)+2 - \left(\frac{n-k_\alpha}{\alpha}-1\right)\left(k_\alpha + \frac{n-k_\alpha}{2} - \alpha\right) \\ &= (n+1-\alpha)(n-1)+2 - \frac{(n-k_\alpha-\alpha)(n+k_\alpha-2\alpha)}{2\alpha} \\ &= (n+1-\alpha)(n-1)+2 - \frac{n^2}{2\alpha} + \frac{3n}{2} + \frac{k_\alpha^2}{2\alpha} - \frac{k_\alpha}{2} - \alpha.\end{aligned}$$

Hence, by abuse of notation, if we let $0 \le k_\alpha \le \alpha - 1$ we have that

$$\left|\mathcal{L}_1\left(\boldsymbol{x}^{(\alpha)}\right)\right| = (n+1-\alpha)(n-1)+2 - \frac{n^2}{2\alpha} + \frac{3n}{2} + \frac{k_\alpha^2}{2\alpha} - \frac{k_\alpha}{2} - \alpha,$$

which implies that

$$\begin{aligned}
\mathsf{diff} &= (n+1-\alpha)(n-1)+2-\frac{n^2}{2\alpha}+\frac{3n}{2}+\frac{k_\alpha^2}{2\alpha}-\frac{k_\alpha}{2}-\alpha \\
&\quad -(n+1-(\alpha-1))(n-1)-2+\frac{n^2}{2(\alpha-1)}-\frac{3n}{2} \\
&\quad -\frac{k_{\alpha-1}^2}{2(\alpha-1)}+\frac{k_{\alpha-1}}{2}+\alpha-1 \\
&= -n+n^2\left(\frac{1}{2(\alpha-1)}-\frac{1}{2\alpha}\right)+\left(\frac{k_\alpha^2}{2\alpha}-\frac{k_\alpha}{2}\right) \\
&\quad +\left(\frac{k_{\alpha-1}}{2}-\frac{k_{\alpha-1}^2}{2(\alpha-1)}\right) \\
&= \frac{n^2}{2\alpha(\alpha-1)}-n+\left(\frac{k_\alpha^2}{2\alpha}-\frac{k_\alpha}{2}\right)+\left(\frac{k_{\alpha-1}}{2}-\frac{k_{\alpha-1}^2}{2(\alpha-1)}\right).
\end{aligned}$$

Let us consider the following distinct cases for the value of n.
Case 1 - If $n = 2\alpha(\alpha-1)$ then $k_\alpha = k_{\alpha-1} = 0$ and

$$\mathsf{diff} = \frac{(2\alpha(\alpha-1))^2}{2(\alpha-1)\alpha}-2\alpha(\alpha-1) = 0.$$

Case 2 - If $n = 2\alpha(\alpha-1)+k$ for some integer $1 \le k \le \alpha-2$ then we have that $k_\alpha = k_{\alpha-1} = k$ and

$$\begin{aligned}
\mathsf{diff} &= \frac{n^2}{2\alpha(\alpha-1)}-n+\left(\frac{k^2}{2\alpha}-\frac{k}{2}\right)+\left(\frac{k}{2}-\frac{k^2}{2(\alpha-1)}\right) \\
&= \frac{n^2}{2\alpha(\alpha-1)}-n+\frac{k^2}{2\alpha}-\frac{k^2}{2(\alpha-1)} \\
&= \frac{n^2}{2\alpha(\alpha-1)}-n-\frac{k^2}{2\alpha(\alpha-1)} \\
&= \frac{(2\alpha(\alpha-1)+k)^2}{2\alpha(\alpha-1)}-2\alpha(\alpha-1)-k-\frac{k^2}{2\alpha(\alpha-1)} \\
&= 2\alpha(\alpha-1)+2k+\frac{k^2}{2\alpha(\alpha-1)}-2\alpha(\alpha-1)-k-\frac{k^2}{2\alpha(\alpha-1)} \\
&= k > 0.
\end{aligned}$$

Case 3 - If $n \ge 2\alpha(\alpha-1)+\alpha-1$, then we first note that for any $0 \le k_{\alpha-1} \le \alpha-2$ we have that $\left(\frac{k_{\alpha-1}}{2}-\frac{k_{\alpha-1}^2}{2(\alpha-1)}\right) \ge 0$ and hence

$$\begin{aligned}
\mathsf{diff} &= \frac{n^2}{2\alpha(\alpha-1)}-n+\left(\frac{k_\alpha^2}{2\alpha}-\frac{k_\alpha}{2}\right)+\left(\frac{k_{\alpha-1}}{2}-\frac{k_{\alpha-1}^2}{2(\alpha-1)}\right) \\
&\ge \frac{n^2}{2\alpha(\alpha-1)}-n+\left(\frac{k_\alpha^2}{2\alpha}-\frac{k_\alpha}{2}\right).
\end{aligned}$$

Define $f : [0, \alpha - 1] \to \mathbb{R}$ by $f(x) \triangleq \frac{x^2}{2\alpha} - \frac{x}{2}$. It is easy to verify that f has a single minimum point at $x = \frac{\alpha}{2}$. Hence,

$$\frac{k_\alpha^2}{2\alpha} - \frac{k_\alpha}{2} = f(k_\alpha) \geq f\left(\frac{\alpha}{2}\right) = \frac{\alpha^2}{4 \cdot 2\alpha} - \frac{\alpha}{2} = -\frac{\alpha}{8}$$

and

$$\mathsf{diff} \geq \frac{n^2}{2\alpha(\alpha-1)} - n - \frac{\alpha}{8}.$$

It holds that

$$\frac{n^2}{2\alpha(\alpha-1)} - n - \frac{\alpha}{8} \geq 0$$

if and only if

$$\begin{aligned} n &\geq \alpha(\alpha-1) + \frac{1}{2}\sqrt{4\alpha^4 - 7\alpha^3 + 3\alpha^2} \\ &= \alpha(\alpha-1) + \sqrt{\alpha^4 - \frac{7}{4}\alpha^3 + \frac{3}{4}\alpha^2} \\ &= \alpha(\alpha-1) + \alpha\sqrt{\alpha^2 - \frac{7}{4}\alpha + \frac{3}{4}} \\ &= \alpha(\alpha-1) + \alpha\sqrt{\left(\alpha - \frac{3}{4}\right)(\alpha-1)}. \end{aligned}$$

Note that

$$\alpha\sqrt{\left(\alpha - \frac{3}{4}\right)(\alpha-1)} < \alpha\left(\alpha - \frac{3}{4}\right),$$

and additionally, it can be verified that for any $\alpha > 1$,

$$2\alpha(\alpha-1) + (\alpha-1) \geq \alpha(\alpha-1) + \alpha\left(\alpha - \frac{3}{4}\right),$$

and thus,

$$\begin{aligned} n &\geq 2\alpha(\alpha-1) + (\alpha-1) \\ &> \alpha(\alpha-1) + \alpha\sqrt{\left(\alpha - \frac{3}{4}\right)(\alpha-1)}, \end{aligned}$$

which implies that $\mathsf{diff} > 0$.

Case 4 - If $n = 2\alpha(\alpha-1) - k$ for some integer $1 \leq k \leq \alpha - 2$ then we have that $k_\alpha = \alpha - k$,

$k_{\alpha-1} = \alpha - 1 - k$, and thus

$$\begin{aligned}
\mathsf{diff} &= \frac{n^2}{2\alpha(\alpha-1)} - n + \left(\frac{(\alpha-k)^2}{2\alpha} - \frac{\alpha-k}{2}\right) \\
&\quad + \left(\frac{\alpha-1-k}{2} - \frac{(\alpha-1-k)^2}{2(\alpha-1)}\right) \\
&= \frac{n^2}{2\alpha(\alpha-1)} - n + \frac{\alpha^2}{2\alpha} - \frac{2\alpha k}{2\alpha} + \frac{k^2}{2\alpha} \\
&\quad + \frac{\alpha-1-k-\alpha+k}{2} - \frac{(\alpha-1)^2}{2(\alpha-1)} \\
&\quad + \frac{2(\alpha-1)k}{2(\alpha-1)} - \frac{k^2}{2(\alpha-1)} \\
&= \frac{n^2}{2\alpha(\alpha-1)} - n + \frac{\alpha}{2} - k + \frac{k^2}{2\alpha} - \frac{1}{2} \\
&\quad - \frac{\alpha-1}{2} + k - \frac{k^2}{2(\alpha-1)} \\
&= \frac{n^2}{2\alpha(\alpha-1)} - n + \frac{k^2}{2\alpha} - \frac{k^2}{2(\alpha-1)} \\
&= \frac{(2\alpha(\alpha-1)-k)^2}{2\alpha(\alpha-1)} - 2\alpha(\alpha-1) + k + \frac{k^2}{2\alpha} - \frac{k^2}{2(\alpha-1)} \\
&= 2\alpha(\alpha-1) - 2k + \frac{k^2}{2\alpha(\alpha-1)} - 2\alpha(\alpha-1) \\
&\quad + k - \frac{k^2}{2\alpha(\alpha-1)} \\
&= -k < 0.
\end{aligned}$$

Case 5 - If $\alpha \leq n \leq 2\alpha(\alpha-1) - (\alpha-1)$ then we first note that for any $0 \leq k_\alpha \leq \alpha - 1$ we have that $\left(\frac{k_\alpha^2}{2\alpha} - \frac{k_\alpha}{2}\right) \leq 0$ and hence

$$\begin{aligned}
\mathsf{diff} &= \frac{n^2}{2\alpha(\alpha-1)} - n + \left(\frac{k_\alpha^2}{2\alpha} - \frac{k_\alpha}{2}\right) + \left(\frac{k_{\alpha-1}}{2} - \frac{k_{\alpha-1}^2}{2(\alpha-1)}\right) \\
&\leq \frac{n^2}{2\alpha(\alpha-1)} - n + \left(\frac{k_{\alpha-1}}{2} - \frac{k_{\alpha-1}^2}{2(\alpha-1)}\right).
\end{aligned}$$

Define $f : [0, \alpha-2] \to \mathbb{R}$ by $f(x) \triangleq \frac{x}{2} - \frac{x^2}{2(\alpha-1)}$. It can be verified that f has a single maximum point at $x = \frac{\alpha-1}{2}$ and hence

$$\begin{aligned}
\frac{k_{\alpha-1}}{2} - \frac{k_{\alpha-1}^2}{2(\alpha-1)} &= f(k_{\alpha-1}) \leq f\left(\frac{\alpha-1}{2}\right) \\
&= \frac{\alpha-1}{4} - \frac{(\alpha-1)^2}{8(\alpha-1)} = \frac{\alpha-1}{8}
\end{aligned}$$

and

$$\mathsf{diff} \leq \frac{n^2}{2\alpha(\alpha-1)} - n + \frac{\alpha-1}{8}.$$

It holds that $\frac{n^2}{2\alpha(\alpha-1)} - n + \frac{\alpha-1}{8} \leq 0$ if and only if

$$\begin{aligned}\alpha(\alpha-1) - \sqrt{\frac{(\alpha-1)^2\alpha(4\alpha-1)}{4}} &\leq n \\ &\leq \alpha(\alpha-1) + \sqrt{\frac{(\alpha-1)^2\alpha(4\alpha-1)}{4}}.\end{aligned}$$

Note that

$$\begin{aligned}\alpha(\alpha-1) - \sqrt{\frac{(\alpha-1)^2\alpha(4\alpha-1)}{4}} &= (\alpha-1)\left(\alpha - \sqrt{\frac{\alpha(4\alpha-1)}{4}}\right) \\ &= (\alpha-1)\left(\alpha - \sqrt{\alpha\left(\alpha-\frac{1}{4}\right)}\right) \\ &\leq (\alpha-1)\left(\alpha - \left(\alpha-\frac{1}{4}\right)\right) \\ &\leq \frac{\alpha-1}{4},\end{aligned}$$

and

$$\begin{aligned}\alpha(\alpha-1) + \sqrt{\frac{(\alpha-1)^2\alpha(4\alpha-1)}{4}} &= (\alpha-1)\left(\alpha + \sqrt{\frac{\alpha(4\alpha-1)}{4}}\right) \\ &= (\alpha-1)\left(\alpha + \sqrt{\alpha\left(\alpha-\frac{1}{4}\right)}\right) \\ &\geq (\alpha-1)\left(\alpha + \left(\alpha-\frac{1}{4}\right)\right) \\ &= 2\alpha(\alpha-1) - \frac{\alpha-1}{4},\end{aligned}$$

and since $\alpha \leq n \leq 2\alpha(\alpha-1) - (\alpha-1)$, we have that $\mathsf{diff} < 0$ as required.

Since α is the number of alternating segments in a sequence of length n, it holds that $n \geq \alpha$. In addition, Case 1 states that for $n = 2\alpha(\alpha-1)$ we have that $\mathsf{diff} = 0$. Furthermore, by combining the results from Case 2 and Case 3 we have that for any $n > 2\alpha(\alpha-1)$ the value of diff is a positive number. Similarly Case 4 and Case 5 prove that for any $\alpha \leq n < 2\alpha(\alpha-1)$ the value of diff is negative. Thus,

$$\mathsf{diff} \geq 0 \iff n \geq 2\alpha(\alpha-1).$$

Chapter 4

The Intersection of Insertion and Deletion Balls

Daniella Bar-Lev, Omer Sabary, Yotam Gershon, and Eitan Yaakobi

Abstract

This paper studies the intersections of insertion and deletion balls. The t-insertion, t-deletion ball of a sequence $\boldsymbol{x}$ is the set of all sequences received by t insertions, deletions to $\boldsymbol{x}$, respectively. While the intersection of either deletion balls or insertion balls has been rigorously studied before, the intersection of an insertion ball and a deletion ball has not been addressed so far. We find the maximum intersection size of any two insertion and deletion balls in the binary case. For the special case of one-insertion and one-deletion balls we find the intersection size for all pair of sequences. Then, we derive the largest and average values of this intersection size. Lastly, we present an algorithm that efficiently computes the intersection of any t_1-insertion ball and t_2-deletion ball.

4.1 Introduction

Studying the size of the deletion and insertion balls as well as their intersections is one of the more intriguing combinatorial problems in the area of coding for synchronization channels. The t-deletion ball of some sequence $\boldsymbol{x}$ is defined to be the set of all sequences that can be derived from $\boldsymbol{x}$ by exactly t deletions. Similarly the t-insertion ball of $\boldsymbol{x}$ is the set of all words that can be received by t insertions to $\boldsymbol{x}$. While it is well known that the insertion balls are regular, that is, the ball size does not depend on the center word $\boldsymbol{x}$, the deletion balls sizes indeed depend on the center $\boldsymbol{x}$ [9]. In particular, the minimum size of the deletion ball is achieved when $\boldsymbol{x}$ consists of a single symbol, while the maximum size is received only for the alternating words.

Studying the intersection of deletion and insertion balls [3, 11, 16] has several applications. For example, the largest size of these intersections provide the solution for the sequence reconstruction problem, which was first studied by Levenshtein [10, 11], for insertion and deletion

channels. These largest sizes correspond to the required minimum number of channels when transmitting a codeword over several deletion or insertion channels. These problems have also connection to the generalized Gilbert-Varshamov bound [7], associative memories [8, 18], and list decoding [4, 12, 17]. One of the motivations to specifically study the intersection of a deletion ball together with an insertion ball originates from a recent problem that was addressed in [13]. Assume a sequence $\boldsymbol{x}$ is transmitted over two channels, where the first introduces deletions while the second only insertions. In order to find the list of all possible transmitted sequences, it is necessary to find the intersection of the insertion ball of the first channel's output and the deletion ball of the second channel's output. While significant progress has been accomplished in studying the intersections of either deletion balls or insertion balls, to the best of our knowledge there is no study that considers together the intersection of insertion and deletion balls of two arbitrary words, which is the goal of this paper. Lastly, we note that studying the intersection of insertion and deletion balls contributes also to studying the balls in the Levenshtein metric and the intersection of these balls [1, 14, 15].

The rest of the paper is organized as follows. Section 4.2 presents the definitions used throughout the paper, a necessary and sufficient condition for the intersection of an insertion ball and a deletion ball to be nonempty, and the problems that will be solved in the paper. In Section 4.3 we study the maximum size of a t_1-insertion ball and a t_2-deletion ball for the binary case. Section 4.4 addresses the case of one-insertion and one-deletion balls. We find the intersection size for all words $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ such that $\boldsymbol{y}_1$ is a subsequence of $\boldsymbol{y}_2$. Based on this result we also derive the largest intersection size and the average size of this intersection. Lastly, in Section 4.5 we present an efficient algorithm to compute the intersection of two insertion and deletion balls. It is shown how to improve upon a naive solution, which computes first the deletion and insertion balls of the two sequences and then find their intersection. Due to the lack of space, some of the proofs in the paper are omitted.

4.2 Preliminaries and Problem Statement

Let Σ_q denote the set of integers $\{0, 1, \ldots, q-1\}$ and for an integer $n \geq 0$, let Σ_q^n be the set of all sequences (words) of length n over the alphabet Σ_q. For a word $\boldsymbol{x} \in \Sigma_q^*$, let $|\boldsymbol{x}|$ denote the length of $\boldsymbol{x}$. For an integer t, $0 \leq t \leq n$, a sequence $\boldsymbol{y} \in \Sigma_q^{n-t}$ is a *t-subsequence* of $\boldsymbol{x} \in \Sigma_q^n$ if $\boldsymbol{y}$ can be obtained from $\boldsymbol{x}$ by deleting t symbols from $\boldsymbol{x}$. That is, there exist $n-t$ indices $1 \leq i_1 < i_2 < \cdots < i_{n-t} \leq n$ such that $y_j = x_{i_j}$, for all $1 \leq j \leq n-t$. We say that $\boldsymbol{y}$ is a *subsequence* of $\boldsymbol{x}$ if $\boldsymbol{y}$ is a t-subsequence of $\boldsymbol{x}$ for some t. Similarly, a sequence $\boldsymbol{y} \in \Sigma_q^{n+t}$ is a *t-supersequence* of $\boldsymbol{x} \in \Sigma_q^n$ if $\boldsymbol{x}$ is a t-subsequence of $\boldsymbol{y}$.

For a sequence $\boldsymbol{x} \in \Sigma_q^n$, let $\boldsymbol{x}_{[i,j]}$ be the subsequence $x_i x_{i+1} \cdots x_j$ and for a set of indices $U \subseteq \{1, \ldots, n\}$, the sequence $\boldsymbol{x}_U$ is the *projection* of $\boldsymbol{x}$ on the indices of U, which is the subsequence of $\boldsymbol{x}$ received by the symbols in the entries of U. For $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^*$, the *Levenshtein distance* between $\boldsymbol{x}$ and $\boldsymbol{y}$, $d_L(\boldsymbol{x}, \boldsymbol{y})$, is the smallest number of insertions and deletions that is required to transform $\boldsymbol{x}$ into $\boldsymbol{y}$.

Definition 4.1. *The* **t-deletion ball** *centred at $\boldsymbol{x} \in \Sigma_q^n$, $D_t(\boldsymbol{x}) \subseteq \Sigma_q^{n-t}$, is the set of all t-subsequences of $\boldsymbol{x}$. The* **t-insertion ball** *centred at $\boldsymbol{x} \in \Sigma_q^n$, $I_t(\boldsymbol{x}) \subseteq \Sigma_q^{n+t}$, is the set of all t-supersequences of $\boldsymbol{x}$.*

For a sequence $\boldsymbol{x} \in \Sigma_q^n$, a *run* of $\boldsymbol{x}$ is a maximal subsequence $\boldsymbol{x}_{[i,j]}$ of identical symbols. The number of runs in $\boldsymbol{x}$ is denoted by $\rho(\boldsymbol{x})$. We say that $\boldsymbol{x}_{[i,j]}$ is an *alternating segment* if $\boldsymbol{x}_{[i,j]}$ is a sequence of alternating distinct symbols $\sigma, \sigma' \in \Sigma_q$. Note that $\boldsymbol{x}_{[i,j]}$ is a *maximal alternating segment* if $\boldsymbol{x}_{[i,j]}$ is an alternating segment and $\boldsymbol{x}_{[i-1,j]}, \boldsymbol{x}_{[i,j+1]}$ are not. For example, for $\boldsymbol{x} = 00110121$, $\rho(\boldsymbol{x}) = 6$ and the four maximal alternating segments are 0, 01, 101, 121.

In this work we study *the insertion-deletion intersection problem*. In this problem we let $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \Sigma_q^*$ and $n \in \mathbb{N}$ such that $|\boldsymbol{y}_1| \leq n \leq |\boldsymbol{y}_2|$ and the goal is to find the set of all words $\boldsymbol{x} \in \Sigma_q^n$ such that $\boldsymbol{x}$ is both, a supersequence of $\boldsymbol{y}_1$ and a subsequences of $\boldsymbol{y}_2$. That is, to find the set

$$\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) \triangleq I_{n-|\boldsymbol{y}_1|}(\boldsymbol{y}_1) \cap D_{|\boldsymbol{y}_2|-n}(\boldsymbol{y}_2).$$

The following lemma states a necessary and sufficient condition for $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) = \emptyset$.

Lemma 4.2. *Let $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \Sigma_q^*$, and let n be an integer such that $|\boldsymbol{y}_1| \leq n \leq |\boldsymbol{y}_2|$. Then, $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) \neq \emptyset$ if and only if $\boldsymbol{y}_1$ is a subsequence of $\boldsymbol{y}_2$.*

Proof. Let $\delta \triangleq |\boldsymbol{y}_2| - |\boldsymbol{y}_1|$. Since $|\boldsymbol{y}_1| \leq |\boldsymbol{y}_2|$, we must perform at least δ insertions in order to transform $\boldsymbol{y}_1$ into $\boldsymbol{y}_2$, and hence $d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) \geq \delta$. Assume $d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) = \delta$ then $\boldsymbol{y}_1$ is obtained by deleting δ symbols from $\boldsymbol{y}_2$. That is, there exists a set of indices $U \subseteq [|\boldsymbol{y}_2|]$ such that $|U| = \delta$ and $(\boldsymbol{y}_2)_{[|\boldsymbol{y}_2|]\setminus U} = \boldsymbol{y}_1$. Let $U' \subseteq U$ be a set of indices such that $|U'| = |\boldsymbol{y}_2| - n$. It can be easily verified that

$$(\boldsymbol{y}_2)_{[|\boldsymbol{y}_2|]\setminus U'} \in \mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$$

and hence, $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) \neq \emptyset$.

For the other direction, assume $d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) > \delta$ and assume to the contrary that there exists $\boldsymbol{x} \in \Sigma_q^n$ such that $\boldsymbol{x} \in \mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$. In this case, $\boldsymbol{x}$ can be obtained from $\boldsymbol{y}_2$ by $|\boldsymbol{y}_2| - n$ deletions and $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{x}$ by $n - |\boldsymbol{y}_1|$ deletions, which is a contradiction since $d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) > \delta$. □

According to Lemma 4.2, it is assumed in the rest of the paper that $\boldsymbol{y}_1$ is a subsequence of $\boldsymbol{y}_2$. The goal of this paper is to study the following problems.

Problem 4.1. *Given integers $n_1 \leq n \leq n_2$, find the maximum intersection size, i.e.,*

$$\max_{\boldsymbol{y}_1 \in \Sigma_q^{n_1}, \boldsymbol{y}_2 \in \Sigma_q^{n_2}} |\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)|.$$

Problem 4.2. *Given integers $n_1 \leq n \leq n_2$, find the expected intersection size for only nonempty intersections,*

$$\mathop{\mathbb{E}}_{\substack{\boldsymbol{y}_1 \in \Sigma_q^{n_1}, \boldsymbol{y}_2 \in \Sigma_q^{n_2} \\ \boldsymbol{y}_1 \text{ is a subsequence of } \boldsymbol{y}_2}} [|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)|].$$

Problem 4.3. *Given $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \Sigma_q^*$ and an integer n such that $|\boldsymbol{y}_1| \leq n \leq |\boldsymbol{y}_2|$, find the size of $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$.*

Problem 4.4. *Given $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \Sigma_q^*$ and an integer n such that $|\boldsymbol{y}_1| \leq n \leq |\boldsymbol{y}_2|$, find efficient algorithms to calculate the set $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$.*

4.3 Maximal Intersection of Binary Insertion and Deletion Balls

In this section, we fully solve Problem 4.1 for the binary case, i.e., the case where $\boldsymbol{y}_1 \in \Sigma_2^{n_1}$ and $\boldsymbol{y}_2 \in \Sigma_2^{n_2}$.

Theorem 4.3. *For integers $0 \leq t_1 \leq n$, $0 \leq t_2$, and $q = 2$ we have that*

$$\max_{\boldsymbol{y}_1 \in \Sigma_2^{n-t_1}, \boldsymbol{y}_2 \in \Sigma_2^{n+t_2}} |\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)| = \sum_{i=0}^{\min\{t_1,t_2\}} \binom{n}{i}.$$

Proof. Let $\boldsymbol{y}_1 \in \Sigma_q^{n-t_1}, \boldsymbol{y}_2 \in \Sigma_q^{n+t_2}$. If $t_1 \geq t_2$, we have that

$$|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)| = |I_{t_1}(\boldsymbol{y}_1) \cap D_{t_2}(\boldsymbol{y}_2)| \leq |D_{t_2}(\boldsymbol{y}_2)| \leq \sum_{i=0}^{t_2} \binom{n}{i},$$

where the last inequality is proven in [5, 9]. To see that the upper bound is tight, let $\boldsymbol{y}_2$ be an alternating segment of length $n + t_2$ and let $\boldsymbol{y}_1$ be the word that is obtained by deleting the last $t_1 + t_2$ symbols of $\boldsymbol{y}_2$. Note that $\boldsymbol{y}_1$ is an alternating segment of length $n - t_1$. By [5], $|D_{t_2}(\boldsymbol{y}_2)| = \sum_{i=0}^{t_2} \binom{n}{i}$. In addition, any deletion in any word can decrease the number of runs by at most 2. Hence, for each $\boldsymbol{x} \in D_{t_2}(\boldsymbol{y}_2)$, $\rho(\boldsymbol{x}) \geq n + t_2 - (t_1 + t_2) = n - t_1$. Note that if the first symbol of $\boldsymbol{x}$ and $\boldsymbol{y}_2$ is different, then the latter inequality is strong. In this case it is possible to show that we can delete t_1 more bits in $\boldsymbol{x}$ in order to receive $\boldsymbol{y}_1$, that is, $\boldsymbol{x}$ is a supersequence of $\boldsymbol{y}_1$ and hence $\boldsymbol{x} \in \mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$, which implies that

$$|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)| = \sum_{i=0}^{t_2} \binom{n}{i}.$$

The case $t_1 < t_2$ is proved in a similar way using the insertion ball of $\boldsymbol{y}_1$ in order to upper bound the size of $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$ (the size of the insertion ball is given in [9]).

□

4.4 The Intersection of 1-Deletion Ball and 1-Insertion Ball

This section is focused on Problem 4.3 and presents an explicit characterization of the intersection of 1-deletion and 1-insertion balls. Using these results, we solve Problems 4.1 and 4.4 for the case where the balls' radius is one.

Since this section is focused on the specific case where $\boldsymbol{y}_1 \in \Sigma_q^{n-1}, \boldsymbol{y}_2 \in \Sigma_q^{n+1}$, and $d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) = 2$, it holds that $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{y}_2$ by exactly 2 deletions. The following lemma states the possible options to receive $\boldsymbol{y}_1$ by 2 deletions from $\boldsymbol{y}_2$.

Lemma 4.4. *$\boldsymbol{y}_1$ can be obtained from $\boldsymbol{y}_2$ either by deleting two symbols from the same run, or by deleting them from two distinct runs, but not both.*

The next definition will be used in characterizing the intersection size $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$.

Definition 4.5. *Let* $\mathcal{R} : \Sigma_q^{n-1} \times \Sigma_q^{n+1} \to \{0,1,2\}$ *be defined as follows,*

$$\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) \triangleq \begin{cases} 0 & d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) > 2 \\ 1 & d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) = 2 \text{ and } \boldsymbol{y}_2 \xrightarrow{1} \boldsymbol{y}_1 \\ 2 & d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) = 2 \text{ and } \boldsymbol{y}_2 \xrightarrow{2} \boldsymbol{y}_1, \end{cases}$$

where $\boldsymbol{y}_2 \xrightarrow{i} \boldsymbol{y}_1$ *denotes the case where* $\boldsymbol{y}_1$ *can be obtained from* $\boldsymbol{y}_2$ *by deletion(s) from* i *run(s).*

Lemma 4.2 states that if $\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 0$, then $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) = \emptyset$. The size of $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$ when $\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 1$ is given in the following lemma.

Lemma 4.6. *For* $\boldsymbol{y}_1 \in \Sigma_q^{n-1}, \boldsymbol{y}_2 \in \Sigma_q^{n+1}$, *if* $\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 1$ *then* $|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)| = 1$.

To analyze the case where $\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 2$ we need to consider whether $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{y}_2$ by deleting two consecutive runs or not. This will be done using the next definition. Note that this definition will be used in Theorem 4.10, which is the main theorem of this section.

Definition 4.7. *Let* $\mathcal{A} : \Sigma_q^{n-1} \times \Sigma_q^{n+1} \to \{0, 1, \ldots, n-1\}$ *be defined as follows. If* $\boldsymbol{y}_1$ *can be obtained from* $\boldsymbol{y}_2$ *by deleting two consecutive symbols and shortening a maximal alternating segment of* m *symbols in* $\boldsymbol{y}_2$ *to a maximal alternating segment of* $m-2$ *symbols in* $\boldsymbol{y}_1$ *then* $\mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) = m-2$, *and* $\mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 0$ *otherwise.*

For example, let $\boldsymbol{y}_1 = 0100010111$ and $\boldsymbol{y}_2 = 0100001010111$. $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{y}_2$ by deleting two consecutive symbols in the highlighted maximal alternating segment and hence $\mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 6 - 2 = 4$.

Lemma 4.8. *Let* $\boldsymbol{y}_1 \in \Sigma_q^{n-1}$ *and* $\boldsymbol{y}_2 \in \Sigma_q^{n+1}$ *be two words. If* $|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)| > 2$ *then* $\mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) > 0$.

Proof. Let $\boldsymbol{y}_1 \in \Sigma_q^{n-1}$, $\boldsymbol{y}_2 \in \Sigma_q^{n+1}$ be two words such that $|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)| > 2$. Lemma 4.6 implies that $\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 2$ and hence, there exist two indices $i < j$ such that $\boldsymbol{y}_1$ is obtained from $\boldsymbol{y}_2$ by deleting one symbol from the i-th run and one symbol from the j-th run of $\boldsymbol{y}_2$. Assume that i is the smallest such integer and that j is the smallest integer with respect to i. Let $D_1(\boldsymbol{y}_2) = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{\rho(\boldsymbol{y}_2)}\}$, where $\boldsymbol{x}_k$ denotes the word obtained by deleting a symbol from the k-th run of $\boldsymbol{y}_2$. It is clear that $\boldsymbol{x}_i, \boldsymbol{x}_j$ are always supersequences of $\boldsymbol{y}_1$, since $\boldsymbol{x}_j$ is obtained by lengthening the run in $\boldsymbol{y}_1$ corresponding to the i-th run in $\boldsymbol{y}_2$, and vice versa. Since $|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)| > 2$, there exists an index $\ell \neq i, j$ such that $\boldsymbol{x}_\ell$ is a supersequence of $\boldsymbol{y}_1$. Note that the minimality of i implies that $\ell > i$. For $1 \leq k \leq \rho(\boldsymbol{y}_2)$, let σ_k be the symbol of the k-th run of $\boldsymbol{y}_2$ and r_k is its length, i.e., $\boldsymbol{y}_2 = \sigma_1^{r_1}\sigma_2^{r_2}\cdots\sigma_{\rho(\boldsymbol{y}_2)}^{r_{\rho(\boldsymbol{y}_2)}}$. Assume to the contrary that $\mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 0$. By definition, the symbols from the i-th and j-th runs are not two consecutive symbols in a maximal alternating segment of length $m \geq 3$ in $\boldsymbol{y}_2$. Hence, we have the following distinct cases:

Case 1: If $i+1<j$, then we have that

$$\boldsymbol{y}_1 = \sigma_1^{r_1}\cdots\sigma_{i-1}^{r_{i-1}}\sigma_i^{r_i-1}\sigma_{i+1}^{r_{i+1}}\cdots\sigma_{j-1}^{r_{j-1}}\sigma_j^{r_j-1}\sigma_{j+1}^{r_{j+1}}\cdots\sigma_{\rho(\boldsymbol{y}_2)}^{r_{\rho(\boldsymbol{y}_2)}},$$
$$\boldsymbol{x}_\ell = \sigma_1^{r_1}\cdots\sigma_{i-1}^{r_{i-1}}\sigma_i^{r_i}\cdots\sigma_\ell^{r_\ell-1}\cdots\sigma_{\rho(\boldsymbol{y}_2)}^{r_{\rho(\boldsymbol{y}_2)}},$$

where σ^0 is defined to be the empty word. Since $\boldsymbol{x}_\ell$ is a supersequence of $\boldsymbol{y}_1$, it can be verified that $\boldsymbol{y}_1$ must be obtained from $\boldsymbol{x}_\ell$ by deleting a symbol from the i-th run (with respect to the order of runs in $\boldsymbol{y}_2$). Hence, $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{x}_i$ by deleting a symbol either from the j-th or the ℓ-th run (with respect to their indices in $\boldsymbol{y}_2$). This implies that the j-th and the ℓ-th run of $\boldsymbol{y}_2$ are combined to a single run in $\boldsymbol{x}_i$. Thus, we have that either $j<i$ or $\ell<i$ which is a contradiction.
Case 2: If $i+1=j$, then $\ell>i,j$ and we have that

$$\boldsymbol{y}_1 = \sigma_1^{r_1}\cdots\sigma_{i-1}^{r_{i-1}}\sigma_i^{r_i-1}\sigma_{i+1}^{r_{i+1}-1}\sigma_{i+2}^{r_{i+2}}\cdots\sigma_{\rho(\boldsymbol{y}_2)}^{r_{\rho(\boldsymbol{y}_2)}},$$
$$\boldsymbol{x}_\ell = \sigma_1^{r_1}\cdots\sigma_{i-1}^{r_{i-1}}\sigma_i^{r_i}\sigma_{i+1}^{r_{i+1}}\cdots\sigma_\ell^{r_\ell-1}\cdots\sigma_{\rho(\boldsymbol{y}_2)}^{r_{\rho(\boldsymbol{y}_2)}}.$$

Let us prove that $\boldsymbol{y}_1$ is obtained from $\boldsymbol{x}_\ell$ by deleting a symbol from the i-th run (with respect to the order of runs in $\boldsymbol{y}_2$) which is a contradiction by the same arguments as in the previous case. If $r_i>1$ then clearly $\boldsymbol{y}_1$ is obtained from $\boldsymbol{x}_\ell$ by deleting a symbol from the i-th run. Otherwise, since $\mathcal{A}(\boldsymbol{y}_1,\boldsymbol{y}_2)=0$, we have that $\sigma_{i-1}\neq\sigma_{i+1}$. If $r_{i+1}>1$ then again, $\boldsymbol{y}_1$ must be obtained from $\boldsymbol{x}_\ell$ by deleting a symbol from the i-th run. Otherwise, $r_{i+1}=1$ which implies that $\sigma_i\neq\sigma_{i+2}$ and we have that

$$\boldsymbol{y}_1 = \sigma_1^{r_1}\cdots\sigma_{i-1}^{r_{i-1}}\sigma_{i+2}^{r_{i+2}}\cdots\sigma_{\rho(\boldsymbol{y}_2)}^{r_{\rho(\boldsymbol{y}_2)}},$$
$$\boldsymbol{x}_\ell = \sigma_1^{r_1}\cdots\sigma_{i-1}^{r_{i-1}}\sigma_i\sigma_{i+1}\cdots\sigma_\ell^{r_\ell-1}\cdots\sigma_{\rho(\boldsymbol{y}_2)}^{r_{\rho(\boldsymbol{y}_2)}}.$$

and since $\sigma_i\neq\sigma_{i+2}$, we conclude again that $\boldsymbol{y}_1$ is obtained from $\boldsymbol{x}_\ell$ by deleting a symbol from the i-th run. □

The last case for $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ is handled in the next lemma.

Lemma 4.9. *Let $\boldsymbol{y}_1\in\Sigma_q^{n-1}$ and $\boldsymbol{y}_2\in\Sigma_q^{n+1}$ be two words. If $\mathcal{R}(\boldsymbol{y}_1,\boldsymbol{y}_2)=2$ and $\mathcal{A}(\boldsymbol{y}_1,\boldsymbol{y}_2)=m-2$ then $\mathsf{ID}(\boldsymbol{y}_1,\boldsymbol{y}_2,n)=m$.*

Proof. By definition, $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{y}_2$ by deleting two consecutive symbols from a maximal alternating segment of length m. Denote by $j,j+1,\ldots,j+m-1$ the indices of the runs that correspond to this maximal alternating segment. Let $D_1(\boldsymbol{y}_2)=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_{\rho(\boldsymbol{y}_2)}\}$, where $\boldsymbol{x}_i$ is obtained from $\boldsymbol{y}_2$ by deleting one symbol from the i-th run of $\boldsymbol{y}_2$. It can be verified that for any $i\in\{j,\ldots,j+m-1\}$, $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{x}_i$ by deleting a symbol from the $(i-1)$-th run or the $(i+1)$-th run of $\boldsymbol{x}_i$ (at least one of them is part of the maximal alternating segment). Hence $\boldsymbol{x}_i\in\mathsf{ID}(\boldsymbol{y}_1,\boldsymbol{y}_2,n)$. Otherwise, $i\notin\{j,\ldots,j+m-1\}$. Assume to the contrary that $\boldsymbol{x}_i$ is a supersequence of $\boldsymbol{y}_1$. By similar arguments to those presented in the proof of Lemma 4.8, it can be concluded that one of the symbols in the i-th run of $\boldsymbol{y}_2$ is part of the same alternating segment as the j-th run, which is a contradiction. □

Theorem 4.10. *For any two words $\boldsymbol{y}_1 \in \Sigma_q^{n-1}$ and $\boldsymbol{y}_2 \in \Sigma_q^{n+1}$, we have that*

$$\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) = \mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) + \mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2).$$

Proof. If $d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) > 2$ then by the definitions of $\mathcal{R}, \mathcal{A}$ we have that $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) = 0$. Otherwise, by Lemma 4.4 it holds that either $\boldsymbol{y}_2 \xrightarrow{1} \boldsymbol{y}_1$ or $\boldsymbol{y}_2 \xrightarrow{2} \boldsymbol{y}_1$. If $\boldsymbol{y}_2 \xrightarrow{1} \boldsymbol{y}_1$ then by Lemma 4.6, $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) = 1$ and by the definitions of $\mathcal{R}, \mathcal{A}$,

$$\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) + \mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 1 + 0 = 1$$

Lastly, if $\boldsymbol{y}_2 \xrightarrow{2} \boldsymbol{y}_1$ then the result follows from Lemma 4.8 and Lemma 4.9. □

Corollary 4.11. *For any two integers $n, q > 1$ we have that*

$$\max_{\boldsymbol{y}_1 \in \Sigma_q^{n-1}, \boldsymbol{y}_2 \in \Sigma_q^{n+1}} |\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)| = n + 1,$$

and the maximum is obtained only when $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ are both alternating sequences consist from the same symbols $\sigma, \sigma' \in \Sigma_q$, and both start with σ.

Note that the latter corollary is a generalization of Theorem 4.3 for any $q \geq 2$, where the balls' radius is one. Lastly, using Theorem 4.10, it is possible to calculate the expected size of the set $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$. The result is given in the next theorem.

Theorem 4.12. *For any two integers $n, q > 1$ we have that*

$$\mathop{\mathbb{E}}_{\substack{\boldsymbol{y}_1 \in \Sigma_q^{n-1}, \boldsymbol{y}_2 \in \Sigma_q^{n+1} \\ \boldsymbol{y}_1 \text{ is a subsequence of } \boldsymbol{y}_2}} [|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)|] = 2 - \frac{q}{1 + (q-1)(n+1) + (q-1)^2 \binom{n+1}{2}}.$$

4.5 Efficient Algorithm Computing the Intersection of Insertion and Deletion Balls

In this section, we address Problem 4.4 and present an efficient algorithm that given $y_1 \in \Sigma_2^{n-t_1}$, $y_2 \in \Sigma_2^{n+t_2}$, and n calculates $|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)|$. A naive method to calculate this intersection is to compute these two balls, i.e., $I_{t_1}(\boldsymbol{y}_1)$ and $D_{t_2}(\boldsymbol{y}_2)$, and then calculate their intersection. However, since the calculation of the balls is done recursively, this approach will introduce high complexity[1]. The algorithm described in this section is based on dynamic programming, and hence works more efficiently.

The following additional definitions will be used throughout the section. A sequence $\boldsymbol{x}$ is called a *common subsequence* of some words $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t$ if $\boldsymbol{x}$ is a subsequence of each one of these t words. The set of all common subsequences of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t$ is denoted by $\mathcal{CS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t)$ and $\mathsf{LCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t)$ denotes the *length of the longest common subsequence*

[1] since the size of $I_{n-|\boldsymbol{y}_1|}(\boldsymbol{y}_1)$ is $\Theta(|\boldsymbol{y}_1|^{n-|\boldsymbol{y}_1|})$ and the largest size of $D_{|\boldsymbol{y}_2|-n}(\boldsymbol{y}_2)$ is $\Theta(|\boldsymbol{y}_2|^{|\boldsymbol{y}_2|-n})$, the worst case complexity of this solution is $\Theta(n^{|\boldsymbol{y}_2|-|\boldsymbol{y}_1|+1}) = \Theta(n^{t_1+t_2+1})$.

(*LCS*) of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t$, that is, $\mathsf{LCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t) = \max_{\boldsymbol{x} \in \mathcal{CS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t)} \{|\boldsymbol{x}|\}$. The set of all longest common subsequences of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t$ is denoted by $\mathcal{LCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t)$.

Let us denote by $\mathcal{U}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ the set of index sets U such that the projection of $\boldsymbol{y}_2$ on the index set U yields $\boldsymbol{y}_1$, that is,

$$\mathcal{U}(\boldsymbol{y}_1, \boldsymbol{y}_2) = \{U \subseteq [|\boldsymbol{y}_2|] : \ (\boldsymbol{y}_2)_U = \boldsymbol{y}_1\}.$$

We say that an index set U is a *right-most index set* of a sequence $\boldsymbol{y}$ if all the indices of U are the right most indices in their run in $\boldsymbol{y}$, i.e., for all $i \in U$, either i is the right most index of its run in $\boldsymbol{y}$, or $i+1 \in U$. Furthermore, denote by $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2) \subseteq \mathcal{U}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ the set of all right-most index sets of $\boldsymbol{y}_2$ in $\mathcal{U}(\boldsymbol{y}_1, \boldsymbol{y}_2)$[2]. We next show how to exhaustively and efficiently [3] scan all vectors in the set $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$. This will be done by first considering the right-most index sets of $\boldsymbol{y}_2$ such that the corresponding projections yields $\boldsymbol{y}_1$ and then complete them with an arbitrary set of indices on the remaining set of indices to receive a length-n word in $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$.

Theorem 4.13. *It holds that*

$$\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) = \left\{(\boldsymbol{y}_2)_{U \cup V} : U \in \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2), V \subseteq [|\boldsymbol{y}_2|] \backslash U, \ |V| = n - |U|\right\}.$$

Proof. Let $U \in \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$. By definition, $(\boldsymbol{y}_2)_U = \boldsymbol{y}_1$. Hence, $\boldsymbol{y}_1$ is a subsequence of $(\boldsymbol{y}_2)_{U \cup V}$. In addition, since $U \cup V \subseteq [|\boldsymbol{y}_2|]$, $\boldsymbol{y}_2$ is a supersequence of $(\boldsymbol{y}_2)_{U \cup V}$. That is, $\boldsymbol{y}_2$ can be obtained from $(\boldsymbol{y}_2)_{U \cup V}$ by $|\boldsymbol{y}_2| - |U \cap V|$ insertions and $\boldsymbol{y}_1$ can be obtained from $(\boldsymbol{y}_2)_{U \cup V}$ by $|V|$ deletions. Namely, that is to say that $(\boldsymbol{y}_2)_{U \cup V} \in \mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$.

For the other direction, let $\boldsymbol{x} \in \mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$ be a sequence. By definition, $\boldsymbol{x}$ is a subsequence of $\boldsymbol{y}_2$. Hence, there is a set of indices $T \subseteq [|\boldsymbol{y}_2|]$ such that $(\boldsymbol{y}_2)_T = \boldsymbol{x}$. Let the number of indices in T that are contained in the i-th run of $\boldsymbol{y}_2$ be denoted by $T(i)$ and let T' be the index set that consists of the $T(i)$ right-most indices of the i-th run of $\boldsymbol{y}_2$ for $1 \leq i \leq \rho(\boldsymbol{y}_2)$. Then. it holds that $(\boldsymbol{y}_2)_{T'} = \boldsymbol{x}$. In addition, since $\boldsymbol{y}_1$ is a subsequence of $\boldsymbol{x} = (\boldsymbol{y}_2)_{T'}$, there is a set of indices $U \subseteq T'$ such that $(\boldsymbol{y}_2)_U = \boldsymbol{y}_1$. Since the indices in T' are the right-most indices of each run of $\boldsymbol{y}_2$, there is an index set $U' \subseteq T'$ that consists of the $U(i)$ right-most indices of the i-th run of $\boldsymbol{y}_2$ for $1 \leq i \leq \rho(\boldsymbol{y}_2)$ and satisfies $(\boldsymbol{y}_2)_{U'} = \boldsymbol{y}_1$. That is, $U' \in \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ and for $V = T' \backslash U'$ we have that $(\boldsymbol{y}_2)_{U' \cup V} = (\boldsymbol{y}_2)_{T'} = \boldsymbol{x}$. □

Using Theorem 4.13 we have the following algorithm for the calculation of the intersection $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$,

Notice that replacing any index in $j \in V$ with an index $j' \in [|\boldsymbol{y}_2|] \backslash (U \cup V)$ from the same run of $\boldsymbol{y}_2$ has no affect on $(\boldsymbol{y}_2)_{U \cup V}$. That is, $(\boldsymbol{y}_2)_{U \cup V} = (\boldsymbol{y}_2)_{U \cup (V \backslash \{j\}) \cup j'}$ for any two indices $j, j' \notin U$ such that $j \in V, j' \notin V$ and j, j' belong to the same run in $\boldsymbol{y}_2$. Hence, in the

[2] Not to be confused with the right canonical embedding (also called canonical embedding) presented in [2]. The right canonical embedding is defined as the embedding that consists of the largest possible indices, and note that the canonical embedding is unique.

[3] It is possible to show that the expectation of $|\mathcal{U}_{right}|$ is upper bounded by $O(n^{\min\{t_1, t_2\}})$. Therefore, for fixed values of t_1 and t_2, the average time complexity of Algorithm 1 is $O(n^{\min\{t_1, t_2\}})$.

Algorithm 1: $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$

Input: $\boldsymbol{y}_1, \boldsymbol{y}_2, n$
Output: $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$
Calculate $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$
Set $S = \emptyset$
for *each* $U \in \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ **do**
  for *each* $V \subseteq [|\boldsymbol{y}_2|] \backslash U$ *such that* $|V| = n - |U|$ **do**
    Calculate $\boldsymbol{x} = (\boldsymbol{y}_2)_{U \cup V}$
    Set $S = S \cup \{\boldsymbol{x}\}$
  end
end
return S

second for loop of Algorithm 1, it is sufficient to iterate only over indices sets V that differ in the number of indices from the same run.

It is left to calculate the sets of index sets $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$. The algorithm to calculate $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ is based on the dynamic programming implementation of the LCS problem [6], which is shortly explained next. Given two words $\boldsymbol{x}, \boldsymbol{y}$, let $\mathsf{LCS}(i, j)$ denote the length of the LCS of $\boldsymbol{x}_{[i]}$ and $\boldsymbol{y}_{[j]}$. The length of the LCS of $\boldsymbol{x}$ and $\boldsymbol{y}$ is given by $\mathsf{LCS}(|\boldsymbol{x}|, |\boldsymbol{y}|)$ and is computed using the following recursive formula

$$\mathsf{LCS}(i, j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ 1 + \mathsf{LCS}(i-1, j-1) & \boldsymbol{x}(i) = \boldsymbol{y}(j) \\ \max\{\mathsf{LCS}(i-1, j), \mathsf{LCS}(i, j-1)\} & \text{otherwise} \end{cases}$$

The implementation of this computation is done using a $(|\boldsymbol{x}| + 1) \times (|\boldsymbol{y}| + 1)$ matrix, which is referred as the *dynamic programming table*, where the j-th entry of the i-th row of the dynamic programming table quals to $\mathsf{LCS}(i, j)$.

The calculation of the set of index sets in $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ will also be done using the $(|\boldsymbol{y}_1| + 1) \times (|\boldsymbol{y}_2| + 1)$ dynamic programming table. Observe that under the problem specifications, $\boldsymbol{y}_1$ is a subsequence of $\boldsymbol{y}_2$, and thus it holds that $\mathcal{LCS}(\boldsymbol{y}_1, \boldsymbol{y}_2) = \{\boldsymbol{y}_1\}$ and thus $\mathsf{LCS}(|\boldsymbol{y}_1|, |\boldsymbol{y}_2|) = |\boldsymbol{y}_1|$. Therefore, there exists at least one index set U such that $(\boldsymbol{y}_2)_U = \boldsymbol{y}_1$. However, in order to find all such right-most index sets, the idea is to search within the dynamic programming table, in order to identify these index sets U which consist of only the right-most indices of each run of $\boldsymbol{y}_2$ and satisfy $(\boldsymbol{y}_2)_U = \boldsymbol{y}_1$.

In order to search such index sets U within the dynamic programming table in an efficient way, we use some of its properties. First, note that the size of each such set U is $|\boldsymbol{y}_1|$. Let $U = \{i_1, i_2, \ldots, i_{|\boldsymbol{y}_1|}\}$, since $(\boldsymbol{y}_2)_U = \boldsymbol{y}_1$, the j-th entry of the i_j-th column of the dynamic programming table equals to j. By the above, the column that corresponds to the i_j-th index contains the value j in the j-th entry, and the i_j-th index represents the appearance of the j-th symbol of $\boldsymbol{y}_1$ in $\boldsymbol{y}_2$. Since each selected index i_j corresponds to the j-th symbol of $\boldsymbol{y}_1$, which corresponds to the j-th row of the dynamic programming table, the selection of i_j allows us to eliminate the j-th row of the dynamic programming table from the rest of the search. We

choose the indices of U in a backward order, and by doing so each selection of index reduces the number of rows by one in the matrix we need to search in the rest of the algorithm.

Before we present the explicit algorithm to compute the set $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$, let us introduce a few data structures that will be used in the algorithm.

1) A two-dimensional array, called match, such that $\mathsf{match}(i, j) = 1$ if $\boldsymbol{y}_1[i] = \boldsymbol{y}_2[j]$ and otherwise 0.

2) A two-dimensional array LCS, which is the dynamic programming table of the LCS algorithm.

3) A binary vector curr of length $|\boldsymbol{y}_2|$. The set of non-zero entries of curr correspond to the indices in a set $U \in \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ in the current step of the search.

By using the above characterization, we design the following recursive procedure, presented in Algorithm 2, in order to calculate the set $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$. Algorithm 2 is initially invoked with the sequences $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, the indices $i = |\boldsymbol{y}_1|,\ j = |\boldsymbol{y}_2|$, and the all zero vector curr.

Algorithm 2: The calculation of $\mathcal{U}_{right}$

Input: $\boldsymbol{y}_1, \boldsymbol{y}_2, i, j$ and a pointer to curr
$\mathcal{U}_{right} = \emptyset$
if $i = 0$ *or* $j = 0$ **then**
  $\mathcal{U}_{right} \leftarrow \{\mathsf{curr}\} \cup \mathcal{U}_{right}$
  return $\mathcal{U}_{right}$
end
if $\mathsf{LCS}(i, j) < i$ **then**
  return $\mathcal{U}_{right}$
end
if $\mathsf{match}(i, j) = 1$ *and* ($j = n$ *or* $\boldsymbol{y}_2[j] \neq \boldsymbol{y}_2[j-1]$ *or* $\mathsf{curr}[j] = 1$) **then**
  Set $\mathsf{curr}[j-1] \leftarrow 1$
  $\mathcal{U}_{right} \leftarrow \mathcal{U}_{right} \cup$ Algorithm 2$(\mathsf{curr}, \boldsymbol{y}_1, \boldsymbol{y}_2, i-1, j-1)$
  Set $\mathsf{curr}[j-1] \leftarrow 0$
end
if $\mathsf{LCS}(i, j-1) = i$ **then**
  $\mathcal{U}_{right} \leftarrow \mathcal{U}_{right} \cup$ Algorithm 2$(\mathsf{curr}, \boldsymbol{y}_1, \boldsymbol{y}_2, i, j-1)$
end
return $\mathcal{U}_{right}$.

The next example demonstrates the idea of Algorithm 2.

Example 4.14. *Consider the example shown in Fig.* 4.1*, in which* $\boldsymbol{y}_1 = 0010$ *and* $\boldsymbol{y}_2 = 000111010$ *and each table is the dynamic programming table. The four highlighted columns in each table correspond to one set of indices* $U \in \mathcal{U}_{right}$.

Theorem 4.15. *Let* $\boldsymbol{y}_1$ *and* $\boldsymbol{y}_2$ *be two sequences such that* $|\boldsymbol{y}_1| \leq |\boldsymbol{y}_2|$. *The output of Algorithm* 2 *is* $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ *if* $\boldsymbol{y}_1$ *is a subsequence of* $\boldsymbol{y}_2$*, and is* $\emptyset$ *otherwise.*

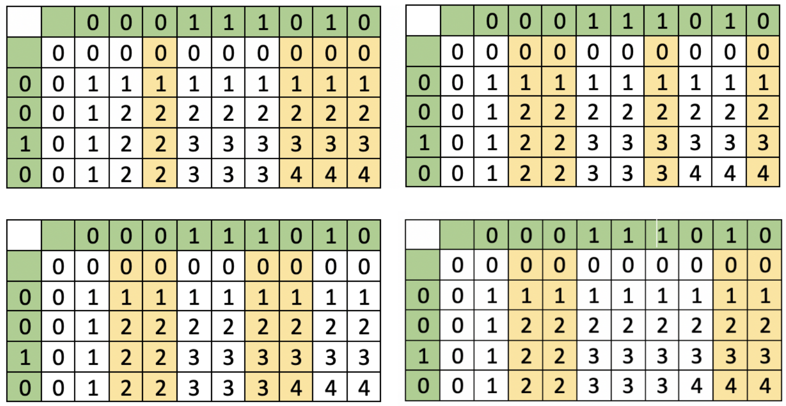


Figure 4.1: An example of Algorithm 2 for $y_1 = 0010$ and $y_2 = 000111010$, so $\mathsf{LCS}(y_1, y_2) = 4$. The highlighted columns represent the right-most index sets $U \in \mathcal{U}_{right}(y_1, y_2)$. More specifically, theses sets, ordered in a clockwise manner, are: $\{3,7,8,9\}$, $\{2,3,6,9\}$, $\{2,3,8,9\}$, $\{2,3,6,7\}$.

Proof. Let $\mathcal{U}_{right}$ denote the output of Algorithm 2 for an input of y_1 and y_2, i.e.,

$$\mathcal{U}_{right} = \text{Algorithm 2}(\mathsf{curr}, y_1, y_2, |y_1|, |y_2|).$$

If y_1 is not a subsequence of y_2 it must be that $1 \leq |y_1|$, and by the assumption, $|y_1| \leq |y_2|$. Thus, the first if condition does not hold. Also, it must be that $\mathsf{LCS}(|y_1|, |y_2|) < |y_1|$, so the algorithm output is $\mathcal{U}_{right} = \emptyset$.

Let us assume that y_1 is a subsequence of y_2. First, we will prove that $\mathcal{U}_{right} \subseteq \mathcal{U}_{right}(y_1, y_2)$. At the initialization $i \leq j$ and in each recursive call we can decrease both i and j by one or only decrease j by one. Since the latter is possible only if

$$i = \mathsf{LCS}(i, j-1) \leq \min\{i, j-1\} \leq j-1,$$

it holds that $i \leq j$ throughout the run of the algorithm. For convenience, we consider the vector curr to be the index set $U := \{j \mid \mathsf{curr}[j-1] = 1\}$ it represents. An index set U is inserted to $\mathcal{U}_{right}$ only when the first if condition holds, that is, only if $i = 0$ or $j = 0$ and since $i \leq j$, that is, whenever $i = 0$. By the definition of the algorithm, i and U can only be changed when the third if condition holds. In this case, j is inserted to U and i and j are decreased by one. Thus, if $i = 0$, U must contain $|y_1|$ indices $j_1, \ldots, j_{|y_1|}$ and by the third condition each j_i satisfies $(y_1)_i = (y_2)_{j_i}$. Therefore, each index set U in the output satisfies $y_1 = (y_2)_U$ which implies that $\mathcal{U}_{right} \subseteq \mathcal{U}(y_1, y_2)$.

Let $U \in \mathcal{U}_{rignt}$, and denote by $j_1 < j_2 < \cdots < j_{|y_1|}$ the indices in U. Since $(y_2)_U = y_1$, for $1 \leq i \leq |y_1|$ it holds that $(y_2)_{\{j_i, \ldots, j_{|y_1|}\}} = (y_1)_{[i\,:\,|y_1|]}$. We will prove, by backward induction that $\{j_1, \ldots, j_{|y_1|}\}$ is a right-most index set. For $i = |y_1|$, if $j_{|y_1|} = |y_2|$ it is clearly a right-most index set. Otherwise, due to the fact that $j_{|y_1|}$ was inserted into U in the third if condition of the algorithm, it must be that $(y_2)_{j_{|y_1|}} \neq (y_2)_{j_{|y_1|}+1}$ (it cannot be that $j_{|y_1|} + 1 \in U$ since U is empty). Thus, $j_{|y_1|}$ is the right-most index in its run in y_2 and

n	The Naive Algorithm	Algorithm 1
50	1558	6.37
75	10449	6.79
100	44652	8.38
125	> 88000	9.46

Table 4.1: A comparison of the run time (in seconds) of Algorithm 1 and the naive algorithm to compute $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$.

$\{j_{|\boldsymbol{y}_1|}\}$ is a right-most index set. Let us assume that $\{j_{i+1}, \ldots, j_{|\boldsymbol{y}_1|}\}$ is a right-most index. Similarly to the base case, j_i was inserted into U in the third if condition. Hence, it must be that $(\boldsymbol{y}_2)_{j_i} \neq (\boldsymbol{y}_2)_{j_i+1}$ or that $j_i + 1 \in U'$ where U' is the subset of U in the current step of the algorithm. By the induction assumption $\{j_{i+1}, \ldots, j_{|\boldsymbol{y}_1|}\}$ is a right-most index set and if $(\boldsymbol{y}_2)_{j_i} \neq (\boldsymbol{y}_2)_{j_i+1}$, then j_i is the right-most index in its run and $\{j_i, j_{i+1}, \ldots, j_{|\boldsymbol{y}_1|}\}$ is also a right-most index set. Otherwise, j_i and $j_i + 1$ belong to the same run of $\boldsymbol{y}_2$ and $j_i + 1 \in U'$. By the induction assumption all the indices that are greater than $j_i + 1$ and belong to the same run as $j_i + 1$ are also in U', that is, all the indices that are right to j_i and at the same run are in U' and $\{j_i, j_{i+1}, \ldots, j_{|\boldsymbol{y}_1|}\}$ is a right-most index set by definition. We have proven that each index set $U \in \mathcal{U}_{right}$ is a right-most index set, that is, we have proven that $\mathcal{U}_{right} \subseteq \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$.

To see that $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2) \subseteq \mathcal{U}_{right}$ consider some index set $U \in \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ and let $j_1 < \cdots < j_{|\boldsymbol{y}_1|}$ be the indices in U. Since $U \in \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$, for each i and $j \geq j_i$ we have that $\mathsf{LCS}(i, j) = i$, that is, the fourth condition holds. Consider the path that invokes the first recursive call if and only if $j = j_i$ for some index $1 \leq i \leq |\boldsymbol{y}_1|$, and invokes the second recursive call otherwise. Since U is a right-most index set, one can easily verify that this is a valid path of the algorithm that yields the index set U and append it to the algorithm output U_{right}. □

We used simulations to evaluate the performance of Algorithm 1. Our simulations worked on 4 different values of $n = \{50, 75, 100, 125\}$. For each n, we created $5,000$ test cases as follows. First, the values of t_1 and t_2 were generated according to the standard normal distribution with mean $\mu = 4$ and standard deviation of $\sigma = 0.5$. Next, $\boldsymbol{y}_2$ was selected randomly from all the sequences of length $n + t_2$. Then, $t_2 + t_1$ bits were selected randomly and deleted from $\boldsymbol{y}_2$ to create $\boldsymbol{y}_1$. We then performed Algorithm 1 and the naive algorithm (described in the begining of this section) to compute $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$ and evaluated their run time. Both algorithms were implemented in $c++$ and performed on our server with Intel(R) Xeon(R) CPU E5-2630 v3 2.40GHz. In all of the tests Algorithm 1 performed the computation of $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$ significantly faster and improve the speed of the naive algorithm by a factor of more than $5,000$. The results of the tests are summarized in Table 4.1. Note that we performed the comparison over relatively small values of the parameters due to the limitations of the naive algorithm.

Acknowledgment

The authors thank Avital Boruchovsky for his helpful comments on the proof of Lemma 4.8. This work was supported in part by the United States-Israel BSF grant no. 2018048.

Bibliography

[1] D. Bar-Lev, T. Etzion, and E. Yaakobi, "On Levenshtein balls with radius one," *IEEE International Symposium on Information Theory*, Melbourne, Australia, Jul. 2021, pp. 1979–1984.

[2] C. Elzinga, S. Rahmann, and H. Wang. "Algorithms for subsequence combinatorics," *Theoretical Computer Science*, vol. 409, no. 3, pp. 394–404, Dec. 2008.

[3] R. Gabrys and E. Yaakobi, "Sequence reconstruction over the deletion channel", *IEEE Transactions on Information Theory*, vol. 64, no. 4, pp. 2924–2931, Apr. 2018.

[4] T. Hayashi and K. Yasunaga, "On the list decodability of insertions and deletions," *IEEE International Symposium on Information Theory*, vol. 66, no. 9, pp. 5335–5343, 2020.

[5] D. S. Hirschberg, "Bounds on the number of string subsequences," *Annual Symposium on Combinatorial Pattern Matching*, 1999, pp. 115–122.

[6] S. Y. Itoga, "The string merging problem," *BIT Numerical Mathematics*, vol. 21, no. 1, pp. 20–30, 1981.

[7] T. Jiang and A. Vardy, "Asymptotic improvement of the Gilbert-Varshamov bound on the size of binary codes," *IEEE Transactions on Information Theory*, vol. 50, no. 8, pp. 1655–1664, Aug. 2004.

[8] V. Junnila, T. Laihonen, and T. Lehtilä, "The Levenshtein's channel and the list size in information retrieval," *2019 IEEE International Symposium on Information Theory*, 2019, pp. 295–299.

[9] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics. Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[10] V. I. Levenshtein, "Efficient reconstruction of sequences," *IEEE Transactions on Information Theory*, vol. 47, no. 1, pp. 2–22, Jan. 2001.

[11] V. I. Levenshtein, "Efficient reconstruction of sequences from their subsequences or supersequences," *Journal of Combinatorial Theory. Series A*, vol. 93, no. 2, pp. 310–332, 2001.

[12] S. Liu, I. Tjuawinata, and C. Xing, "On list decoding of insertion and deletion errors," https://arxiv.org/abs/1906.09705, 2019.

[13] O. Sabary, E. Yaakobi, and A. Yucovich, "The error probability of maximum-likelihood decoding over two deletion/insertion channels," *2020 IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 763–768.

[14] F. Sala and L. Dolecek, "Counting sequences obtained from the synchronization channel," *2013 IEEE International Symposium on Information Theory (ISIT)*, 2013, pp. 2925–2929.

[15] F. Sala, R. Gabrys, and L. Dolecek, "Gilbert-Varshamov-like lower bounds for deletion-correcting codes," *2014 IEEE Information Theory Workshop (ITW)*, 2014, pp. 147–151.

[16] F. Sala, R. Gabrys, C. Schoeny, and L. Dolecek, "Exact reconstruction from insertions in synchronization codes," *IEEE Transactions on Information Theory*, vol. 63, no. 4, pp. 2428–2445, Apr. 2017.

[17] A. Wachter-Zeh, "List decoding of insertions and deletions," *IEEE Transactions on Information Theory*, vol. 64, no. 9, pp. 6297–6304, Sept. 2018.

[18] E. Yaakobi and J. Bruck, "On the uncertainty of information retrieval in associative memories," *IEEE Transactions on Information Theory*, vol. 65, no. 4, pp. 2155–2165, Apr. 2019.

Part II

Coding for DNA Storage

Chapter 5

Adversarial Torn-Paper Codes

Daniella Bar-Lev, Sagi Marcovich, Eitan Yaakobi, and Y. Yehezkeally

Abstract

We study the *adversarial torn-paper channel*. This problem is motivated by applications in DNA data storage where the DNA strands that carry information may break into smaller pieces which are received out of order. Our model extends the previously researched probabilistic setting to the worst-case. We develop code constructions for any parameters of the channel for which non-vanishing asymptotic rate is possible and show our constructions achieve asymptotically optimal rate while allowing for efficient encoding and decoding. Finally, we extend our results to related settings included multi-strand storage, presence of substitution errors, or incomplete coverage.

5.1 Introduction

High density and extreme longevity make DNA an appealing medium for data storage, especially for archival purposes [4, 9, 10, 36]. Advances in DNA synthesis and sequencing technologies and recent proofs of concept [5, 9, 13, 16, 17, 25] have ignited active research into the capacity and challenges of data storage in this medium.

An aspect of this medium is that typically only short DNA sequences may be read; information molecules are therefore broken up into pieces and then read out of order, such as in shotgun sequencing [6, 14, 23, 27]. Multiple channel models have recently been suggested and studied based on this property. An assumption of overlap in read substrings and (near) uniform coverage leads to the problem of string reconstruction from substring composition [3, 6, 15, 22, 23, 29, 31]; on the contrary, assuming no overlap in read substrings leads to the *torn-paper problem* [24, 26, 32], a problem closely related to the shuffling channel [18, 19, 30, 35]. This problem is motivated by DNA-based storage systems, where the information is stored in synthesized strands of DNA molecules. However, during and after synthesis, the DNA strands may break into smaller segments and due to the lack of ordering among the strands in these systems, all broken segments can only be read out of order [32].

Thus, the goal is to successfully retrieve the data from this collection of read segments of the broken DNA strands.

In the *torn-paper channel* [26,32], also known as the *chop-and-shuffle channel* [24], a long information string is segmented into non-overlapping substrings and their length has some known distribution. The channel outputs an unordered collection of these substrings, preserving their left-to-right orientation. Given the lengths' distribution, the goal is to determine the channel capacity and devise efficient coding techniques. The geometric distribution was first studied in [32], and later in [24] using the Varshamov-Tenengolts (VT) codes [34]. Subsequently, [26] considered almost arbitrary distributions while, additionally, extending the problem by introducing incomplete coverage, i.e., assuming some of the substrings are deleted with some probability.

The torn-paper channel was studied so far only in the probabilistic setting. The goal of this paper is to extend this channel to the worst case, referred to herein as the *adversarial torn-paper channel*. Namely, it is assumed that an information string is adversarially segmented into non-overlapping substrings, where the length of each substring is between $L_{\min}$ and $L_{\max}$, for some given $L_{\min}$ and $L_{\max}$. We show that the capacity of this channel is determined by $L_{\min}$, whereas the capacity of the probabilistic channel was shown to depend on the *average* substring length; nevertheless, we choose this adversarial model here for ease of analysis, and observe that under this setting the average substring length might indeed approach $L_{\min}$. For further discussion of an average-restricted adversary, see Section 5.5.

We study the noiseless adversarial torn-paper channel for a single information string, as well as multiple strings, which is motivated by DNA sequencing technologies where multiple strings are sequenced simultaneously [8, 21, 28]. We also extend the model to either allow for substitution errors affecting the information string prior to segmentation, or for incomplete coverage due to deletion of several segments after the segmentation. In all cases we investigate the values of $L_{\min}$ and $L_{\max}$ that permit codes with non-vanishing asymptotic rates, and develop constructions of codes with efficient encoding and decoding algorithms, asymptotically achieving optimal rates.

The rest of this paper is organized as follows. In Section 5.2, the definitions and notations that will be used throughout the paper are presented, as well as a lower bound on $L_{\min}$ required for the existence of codes for the adversarial torn-paper channel with non-vanishing asymptotic rates. In Section 5.3 we first study the application of a known code construction to the adversarial channel, and observe its limitations in that setting; then, we present the basic construction used throughout the paper for the noiseless case of the single-strand adversarial torn-paper channel, and extend it to the multi-strand case. In Section 5.4 we extend our construction to two noisy settings, including substitution errors or incomplete coverage. We conclude with a summary and remarks in Section 5.5.

5.2 Definitions and Preliminaries

Let Σ be a finite alphabet of size q. For convenience of presentation, we assume Σ is equipped with a ring structure, and in particular identify elements $0, 1 \in \Sigma$. For a positive integer n, let $[n]$ denote the set $[n] \triangleq \{0, 1, \ldots, n-1\}$. Let Σ^* denote the set of all finite strings over

Σ. The length of a string $\boldsymbol{x} \in \Sigma^*$ is denoted by $|\boldsymbol{x}|$. We also denote, for $\boldsymbol{x} = (x_i)_{i\in[n]} \in \Sigma^n$, its *support* $\mathrm{supp}(\boldsymbol{x}) \triangleq \{i \in [n] : x_i \neq 0\}$, and $\|\boldsymbol{x}\| \triangleq |\mathrm{supp}(\boldsymbol{x})|$. For strings $\boldsymbol{x}, \boldsymbol{y} \in \Sigma^*$, we denote their concatenation by $\boldsymbol{x} \circ \boldsymbol{y}$. We say that $\boldsymbol{v}$ is a *substring*, or *segment*, of $\boldsymbol{x}$ if there exist strings $\boldsymbol{u}, \boldsymbol{w}$ (perhaps empty) such that $\boldsymbol{x} = \boldsymbol{u} \circ \boldsymbol{v} \circ \boldsymbol{w}$. If $|\boldsymbol{v}| = \ell$, we specifically say that $\boldsymbol{v}$ is an *ℓ-substring* (*ℓ-segment*) of $\boldsymbol{x}$. If $|\boldsymbol{u}| = i$ then it is said that $\boldsymbol{v}$ is the substring (similarly, ℓ-substring) of $\boldsymbol{x}$ at *location* i. We say that $\boldsymbol{v}$ appears *cyclically* in $\boldsymbol{x}$, at location i, if $\boldsymbol{x} = \boldsymbol{u} \circ \boldsymbol{w}$ and $\boldsymbol{v}$ is the substring of $\boldsymbol{w} \circ \boldsymbol{u}$ at location $(i - |\boldsymbol{u}|)$. For example, 010 is the 3-substring of 00101 at location 1, and also its 3-substring at location 3, where the latter is a cyclic appearance. We avoid using the term *index* as it is reserved to elements of presented constructions.

In our setting, information is stored in an unordered collection of strings over Σ; it might be allowed for the same string to appear with multiplicity in the collection, which is encapsulated in the following formal definition:

$$\mathcal{X}_{n,k} \triangleq \{S = \{\!\{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{k-1}\}\!\} : \forall i, \boldsymbol{x}_i \in \Sigma^n\}.$$

Here, $\{\!\{a, a, b, \ldots\}\!\}$ denotes a multiset; i.e., elements appear with multiplicity (but no order). Note that $|\mathcal{X}_{n,k}| = \binom{k+q^n-1}{k}$. It is assumed that a message $S \in \mathcal{X}_{n,k}$ is read by segmenting all elements of S into non-overlapping substrings of lengths between some fixed values $L_{\min}$ and $L_{\max}$, and all segments are received, possibly with multiplicity, without order or information on which element they originated from. More formally, a *segmentation* of the string $\boldsymbol{x}$ is a multiset $\{\!\{\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{m-1}\}\!\}$, where $\boldsymbol{x}$ can be presented as $\boldsymbol{x} = \boldsymbol{u}_0 \circ \boldsymbol{u}_1 \circ \cdots \circ \boldsymbol{u}_{m-1}$. In case $L_{\min} \leq |\boldsymbol{u}_i| \leq L_{\max}$ for $0 \leq i < m-1$ and $|\boldsymbol{u}_{m-1}| \leq L_{\max}$, then the segmentation is called an *$(L_{\min}, L_{\max})$-segmentation*. The set of all $(L_{\min}, L_{\max})$-segmentations of $\boldsymbol{x}$ is denoted by $\mathcal{T}_{L_{\min}}^{L_{\max}}(\boldsymbol{x})$ and is referred as the *$(L_{\min}, L_{\max})$-segmentation spectrum of $\boldsymbol{x}$*. For example,

$$\mathcal{T}_2^3(00101) = \Big\{\{\!\{001, 01\}\!\}, \{\!\{00, 101\}\!\}, \{\!\{00, 10, 1\}\!\}\Big\}.$$

These definitions are naturally extended for a multiset $S \in \mathcal{X}_{n,k}$, so a *segmentation* of S is a union (as a multiset) of segmentations of all the strings in S (and the same holds for an $(L_{\min}, L_{\max})$-segmentation), and $\mathcal{T}_{L_{\min}}^{L_{\max}}(S)$, the *$(L_{\min}, L_{\max})$-segmentation spectrum of S*, is the set of all $(L_{\min}, L_{\max})$-segmentations of S.

Note that our channel model only restricts the length of the last segment to be at most $L_{\max}$. Such a relaxation is motivated in applications where segmentation of the strings occurs sequentially, so that it might happen that the last segment is shorter than $L_{\min}$, but not larger than $L_{\max}$.

A code $\mathcal{C} \subseteq \mathcal{X}_{n,k}$ is said to be an *$(L_{\min}, L_{\max})$-multistrand torn-paper code* if for all $S, S' \in \mathcal{C}$, $S \neq S'$, it holds that all possible $(L_{\min}, L_{\max})$-segmentations of S, S' are distinct. That is, $\mathcal{T}_{L_{\min}}^{L_{\max}}(S) \cap \mathcal{T}_{L_{\min}}^{L_{\max}}(S') = \emptyset$. For $k = 1$, we simply refer to *$(L_{\min}, L_{\max})$-single strand torn-paper codes*.

In case $L_{\min} = L_{\max} = \ell$, then for convenience, we let $\mathcal{T}_\ell(\boldsymbol{x}) \triangleq \mathcal{T}_\ell^\ell(\boldsymbol{x})$ and $\mathcal{T}_\ell(S) \triangleq \mathcal{T}_\ell^\ell(S)$ and note that in this case $|\mathcal{T}_\ell(\boldsymbol{x})| = |\mathcal{T}_\ell(S)| = 1$. For example, if

$$S = \{\!\{01010, 00101, 11101\}\!\}$$

which may be thought of as a multiset, then

$$\mathcal{T}_2(S) = \Big\{\!\!\Big\{01, 01, 0, 00, 10, 1, 11, 10, 1\Big\}\!\!\Big\}.$$

Note that $\mathcal{T}_\ell(S)$ is only one possible channel output given input S. Nevertheless, $\mathcal{T}_{L_{\min}}(S) \subseteq \mathcal{T}_{L_{\min}}^{L_{\max}}(S)$ for all S and $L_{\min} \leq L_{\max}$, hence every $(L_{\min}, L_{\max})$-multistrand torn-paper code $\mathcal{C} \subseteq \mathcal{X}_{n,k}$ satisfies

$$|\mathcal{C}| \leq |\{\mathcal{T}_{L_{\min}}(S) : S \in \mathcal{X}_{n,k}\}|. \tag{5.1}$$

For all $\mathcal{C} \subseteq \mathcal{X}_{n,k}$ we denote the *rate*, *redundancy* of $\mathcal{C}$ by $R(\mathcal{C}) \triangleq \frac{\log|\mathcal{C}|}{\log|\mathcal{X}_{n,k}|}$, $\mathrm{red}(\mathcal{C}) \triangleq \log|\mathcal{X}_{n,k}| - \log|\mathcal{C}|$, respectively. Throughout the paper, we use the base-q logarithms.

For two non-negative functions f, g of a common variable n, denoting $L \triangleq \limsup_{n\to\infty} \frac{f(n)}{g(n)}$ (in the wide sense, i.e., $L = \infty$ if $\frac{f(n)}{g(n)}$ is unbounded) we say that $f = o_n(g)$ if $L = 0$, $f = \Omega_n(g)$ if $L > 0$, $f = O_n(g)$ if $L < \infty$, and $f = \omega_n(g)$ if $L = \infty$. If f is not positive, we say $f = O_n(g)$ ($f = o_n(g)$) if $|f| = O_n(g)$) (respectively, $|f| = o_n(g)$). We say that $f = \Theta_n(g)$ if $f = \Omega_n(g)$ and $f = O_n(g)$. If clear from context, we omit the subscript from aforementioned notations.

We conclude this section by observing a lower bound on the required segment length $L_{\min}$ for multi-strand torn-paper codes to achieve non-vanishing rates, and in particular rates approaching one.

Lemma 5.1. *If* $\log(k) = o(n)$ *and* $L_{\min} = a\log(nk) + O_{nk}(1)$ *for some* $a \geq 1$*, then*

$$\begin{aligned}&\log|\mathcal{X}_{n,k}| - \log|\{\mathcal{T}_{L_{\min}}(S) : S \in \mathcal{X}_{n,k}\}| \\ &\quad\geq nk\bigg(\frac{1}{a} - a\frac{\log(k)}{n} - O\bigg(\frac{\log\log(nk)}{\log(nk)}\bigg)\bigg).\end{aligned}$$

Proof. First, note that

$$|\mathcal{X}_{n,k}| = \binom{k+q^n-1}{k} \geq \frac{q^{nk}}{k!} \geq \frac{q^{nk}}{k^k},$$

and hence $\log|\mathcal{X}_{n,k}| \geq (n - \log(k))k$. Next, since $|\{\mathcal{T}_{L_{\min}}(S) : S \in \mathcal{X}_{n,k}\}|$ is monotonically non-decreasing in n, we have that

$$\begin{aligned}|\{\mathcal{T}_{L_{\min}}(S) : S \in \mathcal{X}_{n,k}\}| &\leq \binom{k\lceil n/L_{\min}\rceil + q^{L_{\min}} - 1}{q^{L_{\min}} - 1} \\ &\leq \binom{k\lceil n/L_{\min}\rceil + q^{L_{\min}}}{q^{L_{\min}}}.\end{aligned}$$

Now, for $v \geq u \geq 0$ we observe

$$\begin{aligned}\log\binom{u+v}{u} &\leq \log\frac{1}{u!}(u+v)^u \leq u\log\Big(e\Big(1+\frac{v}{u}\Big)\Big) \\ &\leq u\Big(\Big(1+\frac{u}{v}\Big)\log(e) + \log\Big(\frac{v}{u}\Big)\Big) \\ &\leq u\Big(2\log(e) + \log\Big(\frac{v}{u}\Big)\Big),\end{aligned}$$

where we used $\log(1+x) \le \frac{\log(e)}{x} + \log(x)$. Setting $u \triangleq k\lceil n/L_{\min}\rceil \le \frac{nk}{L_{\min}} + k$ and $v \triangleq q^{L_{\min}} = \Theta((nk)^a)$, we have $\frac{v}{u} = \Theta\big((nk)^{a-1}L_{\min}\big) = \Theta\big((nk)^{a-1}\log(nk)\big)$, and therefore $\log\big(\frac{v}{u}\big) = (a-1)\log(nk) + \log\log(nk) + O(1)$. We then conclude

$$\begin{aligned}
\log\big|\{\mathcal{T}_{L_{\min}}(S) : S \in \mathcal{X}_{n,k}\}\big| &\le \left(\frac{nk}{L_{\min}} + k\right)\big((a-1)\log(nk) + \log\log(nk) + O(1)\big) \\
&= (a-1)\frac{nk\log(nk)}{L_{\min}} + k(a-1)\log(nk) \\
&\quad + \left(\frac{nk}{L_{\min}} + k\right)(\log\log(nk) + O(1)) \\
&= (a-1)\frac{nk\log(nk)}{L_{\min}} + k(a-1)\log(nk) + O\left(\frac{nk\log\log(nk)}{\log(nk)}\right) \\
&= nk\Bigg((a-1)\frac{\log(nk)}{L_{\min}} + (a-1)\frac{\log(k)}{n} \\
&\qquad + O\left(\frac{\log(n)}{n}\right) + O\left(\frac{\log\log(nk)}{\log(nk)}\right)\Bigg) \\
&= nk\left(\frac{a-1}{a + O(1/\log(nk))} + (a-1)\frac{\log(k)}{n} + O\left(\frac{\log\log(nk)}{\log(nk)}\right)\right) \\
&= nk\left(\frac{a-1}{a} + (a-1)\frac{\log(k)}{n} + O\left(\frac{\log\log(nk)}{\log(nk)}\right)\right)
\end{aligned}$$

which verifies the lemma's statement. □

We note that throughout this paper, we perform redundancy analysis to the second-most-significant term, and retain the order or magnitude for the remainder; since proofs demonstrate that this asymptotic notation does not in fact hide significant coefficients, we believe this representation is faithful for the purpose of finite-length analysis, as well.

The implications of Lemma 5.1 are more clearly stated in the next corollary.

Corollary 5.2. *Let $\mathcal{C}$ be any $(L_{\min}, L_{\max})$-multistrand torn-paper code. Assuming $\log(k) = o(n)$, if $L_{\min} = (a + o_{nk}(1))\log(nk)$, for some $a \ge 1$, then $R(\mathcal{C}) \le 1 - \frac{1}{a} + o_{nk}(1)$.*

Proof. From (5.1) and Lemma 5.1 we have

$$\begin{aligned}
R(\mathcal{C}) &\le \frac{\log|\{\mathcal{T}_{L_{\min}}(S) : S \in \mathcal{X}_{n,k}\}|}{\log|\mathcal{X}_{n,k}|} \\
&\le 1 - \frac{nk}{\log|\mathcal{X}_{n,k}|}\left(\frac{1}{a} - a\frac{\log(k)}{n} - O\left(\frac{\log\log(nk)}{\log(nk)}\right)\right),
\end{aligned}$$

which, together with $\log|\mathcal{X}_{n,k}| \le nk$, justifies the claim. □

5.3 Constructions of Torn-Paper Codes

In this section we study constructions of torn-paper codes, in context of the bound of Corollary 5.2.

5.3.1 Related Works: Pilot-Based Construction

An explicit and efficient coding scheme was presented in [32] for the probabilistic torn-paper channel. Therein, it was argued that an indexing approach to coding is challenging due to the a priori unknown locations of segmentation by the channel, hence this construction relied on interleaving a *pilot* (or *phase-detection sequence*). We describe this scheme below to study its performance in the adversarial channel.

Construction P. *[32, Sec. VII] Fix an integer $m > 1$. Let n be a multiple of m, to be determined later, and s an integer satisfying $s \geq \log(n/m)$. Let $\boldsymbol{p} \in \Sigma^{n/m}$ be any (n/m)-segment of a de Bruijn sequence [11] of order s, which we refer to as the* pilot.

For $\boldsymbol{x}, \boldsymbol{y} \in \Sigma^{n/m}$, denote $\boldsymbol{x} \perp^s \boldsymbol{y}$ if $\boldsymbol{x}, \boldsymbol{y}$ have no common s-segment, i.e., if for all $i, j \in [n/m - s + 1]$ it holds that $\boldsymbol{x}^{(i)} \neq \boldsymbol{y}^{(j)}$, where $\boldsymbol{x}^{(i)}$ ($\boldsymbol{y}^{(j)}$) is the s-segment of $\boldsymbol{x}$ (respectively, $\boldsymbol{y}$) at location i (respectively, j). Then, we denote $\mathcal{O}_{\boldsymbol{p}} \triangleq \left\{\boldsymbol{c} \in \Sigma^{n/m} : \boldsymbol{c} \perp^s \boldsymbol{p}\right\}$.

For any code $\mathcal{C} \subseteq \Sigma^{n/m}$, we construct a code $\mathcal{C}_{\text{pilot}} \subseteq \Sigma^n$ as follows: for every choice of $m-1$ elements $\left(\boldsymbol{c}_j\right)_{j\in[m-1]} \subseteq \mathcal{C} \cap \mathcal{O}_{\boldsymbol{p}}$ (allowing for repetition), we interleave a single symbol from each $\boldsymbol{p}, \boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_{m-2}$, in order, to construct a codeword $\boldsymbol{c} \in \mathcal{C}_{\text{pilot}}$.

Example 5.3. *Let $q = 2, m = 2, n = 12, s = 3$. We choose* 00010111 *as the binary de Bruijn sequence of order s, and let $\boldsymbol{p} \triangleq$* 000101 *be its (n/m)-prefix. Then,*

$$\begin{aligned}\mathcal{O}_{\boldsymbol{p}} = \{&011100, 011110,\\ &011111, 111100,\\ &111110, 111111\}.\end{aligned}$$

Letting $\mathcal{C} \triangleq \Sigma^5$, and for any choice of $m - 1 = 1$ element of $\mathcal{C} \cap \mathcal{O}_{\boldsymbol{p}} = \mathcal{O}_{\boldsymbol{p}}$, we interleave $\boldsymbol{p}$ with that element to derive the code

$$\begin{aligned}\mathcal{C}_{\text{pilot}} = \{&000101110010, 000101110110,\\ &000101110111, 010101110010,\\ &010101110110, 010101110111\}.\end{aligned}$$

Lemma 5.4. *[32, Sec. VII-B] For all $s \geq \log(n/m)$ it holds that $\mathcal{C}_{\text{pilot}}$ is an $(ms, L_{\max})$-single strand torn-paper code, for any $L_{\max} \geq ms$.*

Proof. We replicate the proof for completeness. Observe that every (ms)-segment $\boldsymbol{u}$ of $\boldsymbol{c} \in \mathcal{C}_{\text{pilot}}$ contains s consecutive symbols from each $\boldsymbol{p}, \boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_{m-2}$; since $\boldsymbol{c}_j \perp^s \boldsymbol{p}$ for every $j \in [m-1]$, the s-segment of $\boldsymbol{p}$ can be uniquely identified. Since $\boldsymbol{p}$ is a segment of a de Bruijn sequence of order s, the location in $\boldsymbol{p}$ of the observed segment can be deduced, and hence the location of $\boldsymbol{u}$ in $\boldsymbol{c}$ can readily be obtained. □

Example 5.5. *Continuing Example 5.3, assume* 010101110110 $\in \mathcal{C}_{\text{pilot}}$ *is passed through an adversarial torn-paper channel with $L_{\min} = ms = 6$ and, say, $L_{\max} = 8$. The received segments are*

$$010101, 110110.$$

Taking the first segment, we decompose the two interleaved strings

$$\bar{\boldsymbol{c}}_0 = 000, \bar{\boldsymbol{c}}_1 = 111;$$

we identify $\bar{\boldsymbol{c}}_0$ *as* the *s-substring of* $\boldsymbol{p}$ *at location* 0*, implying that* $\bar{\boldsymbol{c}}_1$ *is the substring of* $\boldsymbol{c}_0$ *at location* 0*. Similarly, we decompose the second segment into*

$$\tilde{\boldsymbol{c}}_0 = 101, \tilde{\boldsymbol{c}}_1 = 110;$$

since 101 *is the s-substring of* $\boldsymbol{p}$ *at location* 3*, we also have that* $\tilde{\boldsymbol{c}}_1$ *is the substring of* $\boldsymbol{c}_0$ *at location* 3*, i.e.,* $\boldsymbol{c}_0 = 111110 \in \mathcal{O}_{\boldsymbol{p}}$*, confirming* $010101110110 \in \mathcal{C}_{\text{pilot}}$ *was the transmitted sequence.*

For the probabilistic channel studied in [32], $\mathcal{C}$ in Construction P was chosen to be an error-correcting code. Note from the proof of Lemma 5.4 that in our chosen adversarial setting, this is redundant; that element of the construction is preserved in our presentation to support the discussion in Section 5.5 regarding alternate models.

Next, we turn to find the achievable rates of Construction P.

Corollary 5.6. $R(\mathcal{C}_{\text{pilot}}) = \left(1 - \frac{1}{m}\right) R(\mathcal{C} \cap \mathcal{O}_{\boldsymbol{p}})$.

Proof. Observe that $\left|\mathcal{C}_{\text{pilot}}\right| = \left|\mathcal{C} \cap \mathcal{O}_{\boldsymbol{p}}\right|^{m-1} = q^{n\left(1-\frac{1}{m}\right)\log\left(\left|\mathcal{C}\cap\mathcal{O}_{\boldsymbol{p}}\right|\right)/\frac{n}{m}} = q^{n\left(1-\frac{1}{m}\right)R(\mathcal{C}\cap\mathcal{O}_{\boldsymbol{p}})}$. □

The following lemma was implied by [32, Sec. VII-A].

Lemma 5.7. *For all* $\mathcal{C} \subseteq \Sigma^{n/m}$ *there exists* $\boldsymbol{z} \in \Sigma^{n/m}$ *such that*

$$R\big((\boldsymbol{z} + \mathcal{C}) \cap \mathcal{O}_{\boldsymbol{p}}\big) \geq R(\mathcal{C}) - (1 - R(\mathcal{O}_{\boldsymbol{p}})),$$

where $\boldsymbol{z} + \mathcal{C} \triangleq \{\boldsymbol{z} + \boldsymbol{c} : \boldsymbol{c} \in \mathcal{C}\}$.

Proof. Observe that

$$\begin{aligned}
\sum_{\boldsymbol{z}\in\Sigma^{n/m}} \left|(\boldsymbol{z} + \mathcal{C}) \cap \mathcal{O}_{\boldsymbol{p}}\right| &= \sum_{\boldsymbol{z}\in\Sigma^{n/m}} \sum_{\substack{\boldsymbol{c}_1\in\mathcal{C} \\ \boldsymbol{c}_2\in\mathcal{O}_{\boldsymbol{p}}}} \mathbb{1}_{\boldsymbol{z}+\boldsymbol{c}_1=\boldsymbol{c}_2} \\
&= \sum_{\substack{\boldsymbol{c}_1\in\mathcal{C} \\ \boldsymbol{c}_2\in\mathcal{O}_{\boldsymbol{p}}}} \sum_{\boldsymbol{z}\in\Sigma^{n/m}} \mathbb{1}_{\boldsymbol{z}=\boldsymbol{c}_2-\boldsymbol{c}_1} \\
&= \sum_{\substack{\boldsymbol{c}\in\mathcal{C} \\ \boldsymbol{o}\in\mathcal{O}_{\boldsymbol{p}}}} 1 = |\mathcal{C}| \cdot \left|\mathcal{O}_{\boldsymbol{p}}\right|.
\end{aligned}$$

It follows from the pigeonhole principle that there exists $\boldsymbol{z} \in \Sigma^{n/m}$ such that

$$\left|(\boldsymbol{z} + \mathcal{C}) \cap \mathcal{O}_{\boldsymbol{p}}\right| \geq q^{-n/m}|\mathcal{C}| \cdot \left|\mathcal{O}_{\boldsymbol{p}}\right|,$$

which concludes the proof. □

In the rest of the section, it remains to analyze what values of s assure that $1 - R(\mathcal{O}_{\boldsymbol{p}}) = o_n(1)$; we also discuss the implications of these available choices.

Lemma 5.8. *[32, Sec. VII-A] If* $s \triangleq \lceil (2+\delta)\log(n/m) \rceil$ *for some* $\delta > 0$*, then, using* $\mathcal{C} \triangleq \Sigma^{n/m}$ *in Construction P,*

$$\begin{aligned} R(\mathcal{C}_{\text{pilot}}) &\geq 1 - \frac{1}{m} - \frac{m-1}{n} \cdot \frac{1}{(n/m)^{\delta} - 1} \\ &= 1 - \frac{1}{m} - o_n(1). \end{aligned}$$

Proof. Again, we replicate the proof here. Denote for a uniformly chosen $\boldsymbol{c} \in \Sigma^{n/m}$ the event $A_{i,j}$ that $\boldsymbol{c}^{(i)} = \boldsymbol{p}^{(j)}$. Clearly $\Pr(A_{i,j}) = q^{-s}$; using the union bound, $\Pr(\boldsymbol{c} \perp^{s} \boldsymbol{p}) \geq 1 - (n/m)^2 q^{-s} \geq 1 - (n/m)^{-\delta}$, i.e.,

$$\left|\mathcal{O}_{\boldsymbol{p}}\right| \geq q^{n/m}\Big(1 - (n/m)^{-\delta}\Big).$$

It follows from Corollary 5.6 that

$$\begin{aligned} R(\mathcal{C}_{\text{pilot}}) &= \left(1 - \frac{1}{m}\right) R(\mathcal{O}_{\boldsymbol{p}}) \\ &\geq \left(1 - \frac{1}{m}\right)\left(1 + \frac{m}{n}\log\Big(1 - (n/m)^{-\delta}\Big)\right) \\ &\geq 1 - \frac{1}{m} - \frac{(m-1)(n/m)^{-\delta}}{n(1 - (n/m)^{-\delta})}. \end{aligned}$$

□

Unfortunately, Lemma 5.8 doesn't match the upper bound of Corollary 5.2; asymptotically, it produces rate $1 - \frac{2+\delta}{a}$, where $a \triangleq \frac{ms}{\log(n)}$. Further, the construction may only be applied when a is (approximately) an *even integer* ≥ 4. The former can be remedied by replacing the union bound in the analysis of [32, Sec. VII-A] with the Lovász local lemma [33] (similarly to techniques used independently in [38] and [12]), as follows.

Lemma 5.9. *Let* $s \triangleq \lceil \log(n/m) + \log\log(n/m) + \log(3e) \rceil$*. Then, using* $\mathcal{C} \triangleq \Sigma^{n/m}$ *in Construction P,*

$$\begin{aligned} R(\mathcal{C}_{\text{pilot}}) &\geq \left(1 - \frac{1}{m}\right) \cdot \left(1 - \frac{\log(e)}{2\log(n/m)}\right) \\ &= 1 - \frac{1}{m} - O\left(\frac{1}{\log(n)}\right). \end{aligned}$$

Proof. Denote for a uniformly chosen $\boldsymbol{c} \in \Sigma^{n/m}$ the event $A_{i,j}$ that $\boldsymbol{c}^{(i)} = \boldsymbol{p}^{(j)}$. Clearly $p \triangleq \Pr(A_{i,j}) = q^{-s}$, and $A_{i,j}$ is jointly independent of $\left\{A_{i',j'} : |i - i'| \geq s\right\}$, i.e., all except $(n/m - s)(2s-1) - 1 \leq 2sn/m - 1$ distinct events.

For sufficiently large n, observe that

$$\begin{aligned} sq^{-s} &\leq \frac{\log(n/m) + \log\log(n/m) + \log(3e)}{(n/m)\log(n/m)3e} \\ &= \frac{m}{2en} \cdot \frac{2}{3}\left(1 + \frac{\log\log(n/m) + \log(3e)}{\log(n/m)}\right) < \frac{m}{2en}, \end{aligned}$$

where the first inequality is justified by $(s+r)q^{-(s+r)} \leq sq^{-s}$ for $r \geq 0$ and $s \geq \log(e)$. Rearranging, we have $m \geq 2epsn$. Therefore, letting $x \triangleq \frac{ep}{1+ep}$ (hence, $\frac{x}{1-x} = ep$), and recalling for all $x \in (0,1)$ that $1 - x \geq \exp(\frac{-x}{1-x})$, we have

$$\begin{aligned} x(1-x)^{2sn/m-1} &= ep(1-x)^{2sn/m} \\ &\geq p\exp\left(1 - 2epsn/m\right) > p. \end{aligned}$$

It therefore follows from the local lemma that

$$\begin{aligned} \Pr(\boldsymbol{c} \perp^s \boldsymbol{p}) &\geq (1-x)^{(n/m)^2} \geq \exp\left(-ep(n/m)^2\right) \\ &= e^{-e(n/m)^2 q^{-s}} \geq e^{-n/2sm}, \end{aligned}$$

where again we used the fact that $m \geq 2epsn$. That is, $\left|\mathcal{O}_{\boldsymbol{p}}\right| \geq q^{n/m}e^{-n/2sm} = \left(qe^{-1/2s}\right)^{n/m}$, and

$$R(\mathcal{O}_{\boldsymbol{p}}) \geq 1 - \frac{\log(e)}{2s}.$$

Hence, Corollary 5.6 concludes the proof. □

Based on Lemma 5.9, Construction P achieves $1 - \frac{1}{a} - o_n(1)$ rate, where $a \triangleq \frac{ms}{\log(n)}$, asymptotically matching the bound of Corollary 5.2. It also expands the values of a for which the construction may be applied; however, unfortunately a is still restricted to be (approximately) an *integer* ≥ 2. Moreover, encoding $\mathcal{C}_{\text{pilot}}(n)$ involves a choice of $\boldsymbol{p}$, and the authors are not aware of a straightforward way to make this choice while optimizing $R(\mathcal{O}_{\boldsymbol{p}})$; it further requires encoding into (potentially, a sub-code of) $\mathcal{O}_{\boldsymbol{p}}$, which is also, to the best of our knowledge, not readily done in an efficient manner. To bridge that gap, we present in the next section a construction based on an indexing approach, which can be applied for any $a > 1$, asymptotically matching Corollary 5.2 for all choices.

5.3.2 Index-Based Construction

In this section, an index-based construction of single-strand torn-paper codes is presented and is then extended for multiple strands.

It is assumed from here on out that $L_{\min} = \lceil a\log(n) \rceil$, for some $a > 1$ which is fixed throughout this section. We propose the following construction of length-n $(L_{\min}, L_{\max})$-single strand torn-paper codes. The construction is based on the following components.

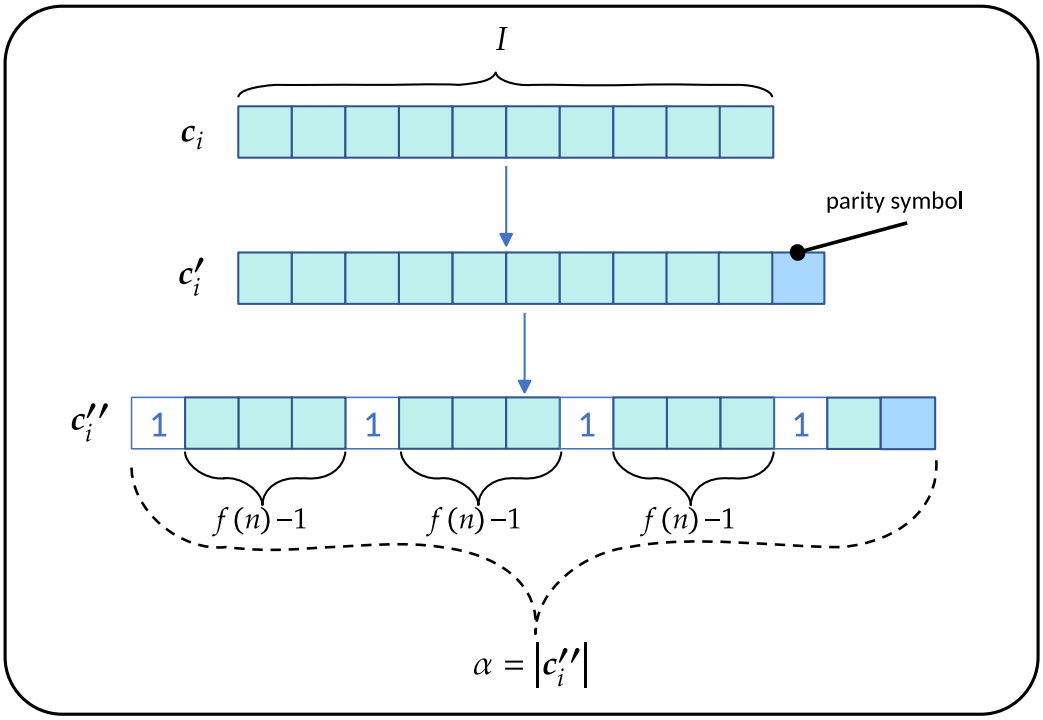


Figure 5.1: Index generation

Definition 5.10. *For an integer I, let $(\boldsymbol{c}_i)_{i\in[q^I]}$, $\boldsymbol{c}_i \in \Sigma^I$ be codewords of a q-ary Gray code, in order. Denote by $\boldsymbol{c}'_i$ the concatenation of $\boldsymbol{c}_i$ with a single parity symbol (i.e., the sum of the entries in $\boldsymbol{c}'_i$ is zero). Further, denote by $\boldsymbol{c}''_i$ the result of inserting '1's into $\boldsymbol{c}'_i$ at every location divisible by $f(n)$ (since the locations of substrings start with 0, the first bit of $\boldsymbol{c}''_i$ is always '1'). The process is illustrated in Figure 5.1. Note that $\alpha \triangleq |\boldsymbol{c}''_i| = \left\lceil \frac{f(n)}{f(n)-1}(I+1) \right\rceil$ for all $i \in [q^I]$. We refer to $\boldsymbol{c}_i$ (or simply i) as an* index *in the construction and to $\boldsymbol{c}''_i$ as an* encoded index.

This index structure is motivated by the property indicated in the following lemma.

Lemma 5.11. *Let $\boldsymbol{c}$ be an α-substring of $\boldsymbol{c}''_i \circ \boldsymbol{c}''_{i+1}$, for some $i \in [q^I - 1]$. Then i can uniquely be recovered from $\boldsymbol{c}$.*

Proof. Since $\boldsymbol{c}''_i$ and $\boldsymbol{c}''_{i+1}$ differ only at the parity symbol and one additional coordinate (which corresponds to the only position where $\boldsymbol{c}_i$ and $\boldsymbol{c}_{i+1}$ differ), $\boldsymbol{c}$ is either $\boldsymbol{c}''_i$ or a copy of $\boldsymbol{c}''_{i+1}$ with an erroneous parity symbol. To obtain i it suffices to distinguish these two cases, which may be done by calculating the parity symbol of $\boldsymbol{c}''$; If the parity symbol is correct then i equals to the decoding of $\boldsymbol{c}$ (with the Gray-code decoder), and otherwise i equals to the decoding of $\boldsymbol{c}$ minus one. □

Definition 5.12. *Let f, N be integers. The* Run-length limited (RLL) encoder E_N^{RLL} *receives strings of length $m(N)$ and returns strings of length N that do not contain zero runs of length f. Constructions of such encoders can be taken from [20] or [37, Lem. 4].*

Algorithm 3: Encoder for Construction A

Input: $\boldsymbol{x} = (x_0, x_1, \ldots, x_{Km(N)-1}) \in \Sigma^{Km(N)}$
Output: $\mathrm{Enc}_A(\boldsymbol{x})$
for $i \leftarrow 0$ *to* $K-1$ **do**
  $\boldsymbol{x}_i \leftarrow (x_{im(N)}, x_{im(N)+1}, \ldots, x_{(i+1)m(N)-1})$ // $|\boldsymbol{x}_i| = m(N)$
  $\boldsymbol{y}_i \leftarrow E_N^{RLL}(\boldsymbol{x}_i)$ // $\boldsymbol{y}_i$ contains no zero runs of length $f(n)$
  $\boldsymbol{z}_i \leftarrow \boldsymbol{c}_i'' \circ 10^{f(n)}1 \circ \boldsymbol{y}_i$ // $|\boldsymbol{z}_i| = L_{\min}$
end
$\boldsymbol{z}_K \leftarrow \boldsymbol{c}_K'' \circ 10^{f(n)}10^N$ // $|\boldsymbol{z}_K| = L_{\min}$
$\boldsymbol{z} \leftarrow \boldsymbol{z}_0 \circ \boldsymbol{z}_1 \circ \cdots \circ \boldsymbol{z}_K \circ 0^{n \bmod L_{\min}}$ // $|\boldsymbol{z}| = n$
return $\boldsymbol{z}$

Construction A. *The main idea of the construction is that every codeword should constitute a concatenation of length-$L_{\min}$ segments with the following structure: an index, followed by a marker, then encoded data. Let $f(n)$ be any integer-valued function satisfying $f(n) = \omega(1)$ and $f(n) = o(\log(n))$ (see Theorem 5.18 for a choice optimizing the redundancy of this construction). Further assume $n \geq L_{\min} \geq \alpha + f(n) + 2$. Let $I \triangleq \lceil \log(n/L_{\min}) \rceil$, $K \triangleq \lfloor n/L_{\min} \rfloor - 1$ and $N \triangleq L_{\min} - \alpha - f(n) - 2$. The constructed $(L_{\min}, L_{\max})$-single strand torn-paper code, denoted by $\mathcal{C}_A(n)$, is defined by the encoder $\mathrm{Enc}_A : \Sigma^{Km(N)} \to \Sigma^n$ in Algorithm 3, and illustrated in Figure 5.2.*

In the rest of the paper, we call the strings $\boldsymbol{x}_i$ (respectively, $\boldsymbol{y}_i$) in the constructions an *information block* (*encoded block*); the strings $10^{f(n)}1$ are called *markers*; finally, a string $\boldsymbol{z}_i$ will simply be referred to as a segment of $\boldsymbol{z}$. Note that the last segment $\boldsymbol{z}_K$ of $\boldsymbol{z}$ deliberately does not contain data, to account for the possibility that a part of $\boldsymbol{z}_K \circ 0^{n \bmod L_{\min}}$ might be partitioned by an adversarial channel in such a way that it does not contain, at its suffix, a prefix of an index. We observe that once the encoded blocks $\boldsymbol{y}_i$'s are obtained, encoding (including the generation of the Gray code) then requires a number of operations linear in n. By [20, 37], encoding each $\boldsymbol{x}_i$ into $\boldsymbol{y}_i$ may also be achieved with a linear number of operations. Hence, the complexity of Construction A is linear with n.

Example 5.13. *We demonstrate the operation of Enc_A. Let $q = 2, n = 45, L_{\min} = 14$, $f(n) = 2$. For index generation, we utilize the binary Gray code $(00, 01, 11, 10)$, whose encoded indices are (in order)*

$$(101010, 101111, 111110, 111011)$$

(observe $\alpha = 6$). Let $N = L_{\min} - f(n) - 2 - \alpha = 4$, and observe an encoder E_N^{RLL} exists with $m(N) = 3$, defined by the lexicographic ordering of the $f(n)$-run-length-limited sequences of length N

$$\{0101, 0110, 0111, 1010, 1011, 1101, 1110, 1111\}.$$

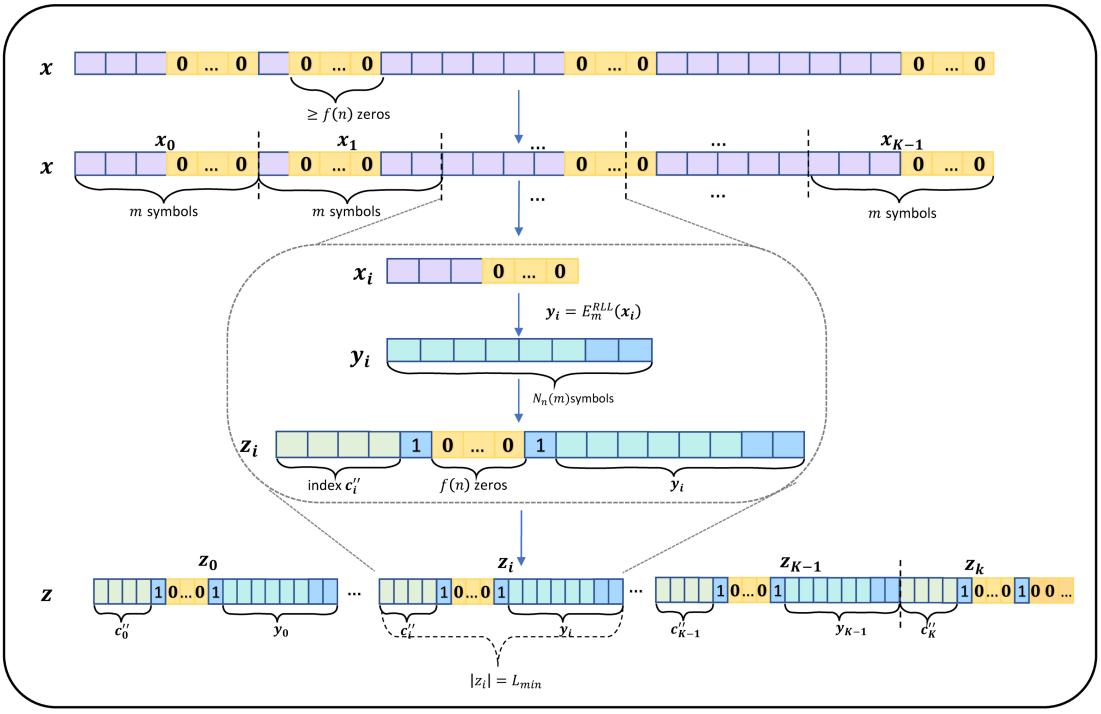


Figure 5.2: Illustration of Algorithm 3

Noting that $K = 2$*, we demonstrate, e.g., the encoding of the information sequence* 001110*. Observe,* $\boldsymbol{x}_0 = 001, \boldsymbol{x}_1 = 110$*, hence* $\boldsymbol{y}_0 = 0110, \boldsymbol{y}_1 = 1110$*. We then have*

$$\begin{aligned} \boldsymbol{z}_0 &= 101010\ 1001\ 0110 \\ \boldsymbol{z}_1 &= 101111\ 1001\ 1110 \\ \boldsymbol{z}_2 &= 111110\ 1001\ 0000, \end{aligned}$$

and $\boldsymbol{z} = \boldsymbol{z}_0 \circ \boldsymbol{z}_1 \circ \boldsymbol{z}_2 \circ 000$*.*

Next, it is shown that the constructed code $\mathcal{C}_A(n)$ is an $(L_{\min}, L_{\max})$-single strand torn-paper code.

Theorem 5.14. *For all* $L_{\max} \geq L_{\min}$*,* $\mathcal{C}_A(n)$ *is an* $(L_{\min}, L_{\max})$*-single strand torn-paper code with a linear-run-time decoder.*

The proof of Theorem 5.14 is carried by presenting an explicit decoder to $\mathcal{C}_A(n)$ as follows. Let $\boldsymbol{z} \in \mathcal{C}_A(n)$ and let $\boldsymbol{z} = \boldsymbol{u}_0 \circ \boldsymbol{u}_1 \circ \cdots \circ \boldsymbol{u}_{s-1}$ so that $\{\!\{\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{s-1}\}\!\}$ is an $(L_{\min}, L_{\max})$-segmentation of $\boldsymbol{z}$. The main task of the decoding algorithm is to successfully retrieve the location within $\boldsymbol{z}$ of each of the s segments of the $(L_{\min}, L_{\max})$-segmentation. For every segment $\boldsymbol{u}_j$, $j \in [s]$, the decoder first finds the location i such that the first (maybe partial) occurrence of an encoded index in the segment $\boldsymbol{u}_j$ is of $\boldsymbol{c}_i''$ (see below for a proof that this is possible). Given i and the location of $\boldsymbol{c}_i''$ in $\boldsymbol{u}_j$, the location of the segment $\boldsymbol{u}_j$ within $\boldsymbol{z}$ can be calculated. Then, according to the location in $\boldsymbol{z}$ for each segment in the $(L_{\min}, L_{\max})$-segmentation, one

Algorithm 4: Index retrieval from a segment

Input: An L-segment $\boldsymbol{u}$ of a codeword of $\mathcal{C}_A(n)$, where $L \geq L_{\min}$.
Output: The index of $\boldsymbol{u}$ within $\boldsymbol{z}$, $\mathrm{Ind}(\boldsymbol{u})$
$\boldsymbol{u}' \leftarrow$ the $L_{\min}$-length prefix of $\boldsymbol{u}$
$j \leftarrow$ the starting index of the unique occurrence of $10^{f(n)}1$ within $\boldsymbol{u}'$; if none exists, of the cyclic occurrence
$\boldsymbol{c}'' \leftarrow$ the (cyclic) α-substring of $\boldsymbol{u}$ strictly preceding j
$\boldsymbol{c}' \leftarrow$ the non-padded subsequence of $\boldsymbol{c}''$
$\boldsymbol{c} \leftarrow$ the I-prefix of $\boldsymbol{c}'$
$\mathrm{Ind} \leftarrow$ the index of $\boldsymbol{c}$ in the Gray code
if *the last symbol of* $\boldsymbol{c}'$ *is not the parity of* $\boldsymbol{c}$ **then**
  $\mathrm{Ind} \leftarrow \mathrm{Ind} - 1$
end
return Ind

can simply concatenate the segments in the correct order to obtain the codeword $\boldsymbol{z}$. Finally, by removing the markers and the encoded indices and applying the RLL decoder for each of the strings $\boldsymbol{y}_i$'s, the information string $\boldsymbol{x}$ is retrieved.

Consider the case where a segment $\boldsymbol{u}$ is a proper substring of the suffix of $\boldsymbol{z}$ of length $(n \bmod L_{\min}) + N + f(n)$, i.e., $\boldsymbol{z}_K 0^{n \bmod L_{\min}}$ (note that this does not imply that $\boldsymbol{u}$ is itself a suffix of $\boldsymbol{z}$). Then, $\boldsymbol{u}$ does not intersect $\boldsymbol{y}_i$ for any $i \in [K]$, and may safely be discarded. We see next that these cases may be identified efficiently.

Lemma 5.15. *Let $\boldsymbol{z} \in \mathcal{C}_A(n)$ and let $\boldsymbol{u}$ be a proper substring of $\boldsymbol{z}_K 0^{n \bmod L_{\min}}$. If n is sufficiently large (specifically, if $(a-1)\lceil \log(n) \rceil > 2f(n)+1$), then this fact can efficiently be identified.*

Proof. Observe that either $|\boldsymbol{u}| < L_{\min}$ or $\boldsymbol{u}$ contains a suffix of '0's of length at least

$$L_{\min} - \alpha - f(n) - 1 \geq (a-1)\lceil \log(n) \rceil - f(n) - 1,$$

i.e., longer than $f(n)$, which can easily be identified. □

By Lemma 5.15, it is sufficient to retrieve the location of any segment which is not a substring of the suffix of length $(n \bmod L_{\min}) + N + f(n)$ of $\boldsymbol{z}$. For any such $\boldsymbol{u}$, the calculation of the index i such that $\boldsymbol{c}_i''$ is the first (perhaps partial) occurrence of an encoded index within $\boldsymbol{u}$, is given in Algorithm 4.

Any L-segment $\boldsymbol{u}$ of $\boldsymbol{z} \in \mathcal{C}_A(n)$, such that $L \geq L_{\min}$, contains at least part of one of the encoded indices $\boldsymbol{c}_i''$. If $\boldsymbol{c}_i''$ is the first encoded index to intersect $\boldsymbol{u}$, we denote by $\mathrm{Ind}(\boldsymbol{u}) \triangleq i$ the *index of* $\boldsymbol{u}$. Note that this index does not depend on the information that was encoded in the construction, but rather, only on the location of $\boldsymbol{u}$ in $\boldsymbol{z}$. Algorithm 4 ensures that it is possible to determine the index of every L-segment $\boldsymbol{u}$ of $\boldsymbol{z}$, where $L \geq L_{\min}$.

The correctness of Algorithm 4 follows from the next lemma.

Lemma 5.16. *Let $\boldsymbol{z} \in \mathcal{C}_A(n)$, $L \geq L_{\min}$, and let $\boldsymbol{u}$ be an L-segment of $\boldsymbol{z}$ which is not a substring of the suffix of length $(n \bmod L_{\min}) + N + f(n)$ of $\boldsymbol{z}$. Then, Algorithm 4 successfully returns the index $\mathrm{Ind}(\boldsymbol{u})$ of $\boldsymbol{u}$.*

Proof. Let $\boldsymbol{u}$ be a substring of $\boldsymbol{z}$ and w.l.o.g. assume that $|\boldsymbol{u}| = L_{\min}$. From the RLL encoding of the strings $\boldsymbol{x}_i$'s, observe that $\boldsymbol{u}$ does not contain any occurrences of $10^{f(n)}1$ except those explicitly added to the encoded indices by Construction A. Since $|\boldsymbol{z}_j| = L_{\min}$ for all j, either $\boldsymbol{u}$ contains an occurrence of $10^{f(n)}1$ or it has a suffix-prefix pair whose concatenation is $10^{f(n)}1$ (this follows from Construction A and the assumption that $\boldsymbol{u}$ does not begin with a proper suffix of $\boldsymbol{z}_K 0^{n \bmod L_{\min}}$). In both cases, we will show that the precise location of the (perhaps incomplete) occurrence of $\boldsymbol{c}''_i$ in $\boldsymbol{u}$ can be deduced, for some i.

Let j be the (unique) location in $\boldsymbol{u}$ of the substring $10^{f(n)}1$. If $j \geq \alpha$, then $\boldsymbol{u}$ contains a complete occurrence of the encoded index $\boldsymbol{c}''_i$, and so the index $\boldsymbol{c}_i$, and therefore i, are readily obtained. Otherwise, $j < \alpha$ and let $\boldsymbol{c}''$ be the cyclic α-substring of $\boldsymbol{u}$ strictly preceding the substring $10^{f(n)}1$ which starts at location $L_{\min} - (\alpha - j)$. The substring $\boldsymbol{c}''$ is obtained by the concatenation of the $(\alpha - j)$-suffix of $\boldsymbol{u}$ with the j-prefix of $\boldsymbol{u}$. The proof is now concluded by Lemma 5.11. □

We remark that the described procedure operates in run-time which is linear in the substring length. In addition, if $\boldsymbol{z}$ can be reconstructed from its non-overlapping substrings, then the strings $\boldsymbol{y}_i$'s are readily obtained, and $\boldsymbol{x}$ may be decoded (again, see [20,37]). These algorithms also require a linear number of operations. This completes the proof of Theorem 5.14.

Example 5.17. *We return to Example 5.13, to demonstrate the operation of Theorem 5.14. Recall, for $q = 2, n = 45, L_{\min} = 14, f(n) = 2$, that we have constructed the following codeword*

$$\boldsymbol{z} = 10101010010110101111100111101111010010000000.$$

Suppose that we receive the following $(14, 20)$-segmentation of $\boldsymbol{z}$:

$$\{\!\{10101010010110101, 1111001111011111, 010010000000\}\!\}.$$

Note since $|010010000000| = 12 < L_{\min}$, it might readily be inferred that it is the suffix of $\boldsymbol{z}$. We therefore only need identify the locations of the other two segments.

- *The segment* 101010 1001 0110101 *contains the marker $10^{f(n)}1 = 1001$, and we therefore conclude that* 101010 *(given $\alpha = 6$) is an encoded-index, which as we recall corresponds to the Gray-code element $c_0 = 00$. It follows that $\boldsymbol{y}_0 = 0110$, and* 101 *is a prefix of $\boldsymbol{z}_1$ (observe that the following segment of $\boldsymbol{z}_1$ could not have been immediately identify, if more segments were received).*

- *Next, the segment* 111 1001 111011111 *also contains a marker, implying that* 111 *is the suffix of c''_i, and* 111 *the prefix of c''_{i+1}. Concatenating, we have the index $\boldsymbol{c}'' = 111111$ and $\boldsymbol{c}' = 11\,1$, which is an instance of $\boldsymbol{c}_2$ containing an erroneous parity symbol. Hence we deduce $i = 1$, and $\boldsymbol{y}_1 = 1110$.*

Together, the decoding $\boldsymbol{x}_0 = 001$ and $\boldsymbol{x}_1 = 110$ may now be performed, reconstructing the original information sequence.

Lastly, the redundancy of Construction A is analyzed.

Theorem 5.18. *Using the RLL encoders of [20, 37] in Construction A, it holds that*

$$\begin{aligned}\operatorname{red}(\mathcal{C}_A(n)) &\leq \frac{n}{a}\Bigg(1+\frac{f(n)}{\log(n)}+\frac{1}{f(n)-1}+\\ &\qquad\frac{9+2/(f(n)-1)}{\log(n)}+\frac{4a}{q^{f(n)}}+\frac{2a^2+2}{n}\Bigg)\\ &= \frac{n}{a}\left(1+(1+o(1))\left(\frac{f(n)}{\log(n)}+\frac{1}{f(n)}\right)\right).\end{aligned}$$

In particular, the redundancy is optimized for $f(n) = (1+o(1))\sqrt{\log(n)}$, i.e.,

$$\operatorname{red}(\mathcal{C}_A(n)) \leq \frac{n}{a}\left(1+\frac{2+o(1)}{\sqrt{\log(n)}}\right).$$

Proof. From Construction A, observe that

$$\operatorname{red}(\mathcal{C}_A(n)) = (n \bmod L_{\min}) + L_{\min} + K(L_{\min} - m(N))$$

and

$$\begin{aligned}L_{\min} - N &= \alpha + f(n) + 2\\ &= \left\lceil \frac{f(n)}{f(n)-1}(I+1)\right\rceil + f(n) + 2\\ &\leq \frac{f(n)}{f(n)-1}(I+1) + f(n) + 3\\ &\leq \frac{f(n)}{f(n)-1}(\log(n/L_{\min})+2) + f(n) + 3\\ &\leq \log(n) + f(n) + \frac{\log(n)}{f(n)-1} + 5 + \frac{2}{f(n)-1}.\end{aligned}$$

Further, by [37, Lem. 4] one may efficiently encode $\boldsymbol{x} \mapsto \boldsymbol{y}$ such that $N - m(N) \leq \left\lceil \frac{q}{q-2} \cdot \frac{N}{q^{f(n)}} \right\rceil$ (For $q = 2$ [20, Sec. III] showed $N - m(N) \leq 2\left\lceil N/q^{f(n)-1}\right\rceil$), and we shall use the overly zealous upper bound $N - m(N) \leq \frac{4N}{q^{f(n)}} + 2 \leq \frac{4a\log(n)}{q^{f(n)}} + \frac{4}{q} + 2 \leq \frac{4a\log(n)}{q^{f(n)}} + 4$.

Finally, we get that

$$\begin{aligned}\operatorname{red}(\mathcal{C}_A(n)) &= K(L_{\min} - m(N)) + L_{\min} + (n \bmod L_{\min})\\ &\leq \frac{n}{a\log(n)}\left(\log(n) + f(n) + \frac{\log(n)}{f(n)-1} + 9 + \frac{2}{f(n)-1} + \frac{4a\log(n)}{q^{f(n)}}\right) + 2L_{\min}\\ &\leq \frac{n}{a}\Bigg(1+\frac{f(n)}{\log(n)}+\frac{1}{f(n)-1}+\quad\frac{9+2/(f(n)-1)}{\log(n)}+\frac{4a}{q^{f(n)}}+\frac{2a^2+2}{n}\Bigg),\end{aligned}$$

which completes the proof of the first part. The second part follows by substitution of $f(n) = (1+o(1))\sqrt{\log(n)}$ into the former. $\square$

By Theorem 5.18 and Corollary 5.2, efficient encoding and decoding is possible at asymptotically optimal rates. In comparison to Construction P (by Lemma 5.9), Construction A asymptotically achieves rate $1 - \frac{1}{a} - O\left(\frac{f(n)}{\log(n)} + \frac{1}{f(n)}\right)$ instead of $1 - \frac{1}{\lceil a \rceil} - O\left(\frac{1}{\log(n)}\right)$, for any channel parameter $a > 1$ (here, the integer value is used since Construction P must be operated at $m \triangleq \lceil a \rceil$ to produce an $(L_{\min}, L_{\max})$-torn-paper code). For completeness, we also include specific construction parameters for several arbitrary choices of $n, L_{\min}$, and compare resulting rates, in Tables 5.1 to 5.3 (all for $q = 4$). It should however be stressed that, for Construction P, the choice of s, m optimizing the resulting rate $R(\mathcal{C}_{\text{pilot}}(n)) \geq \left(1 - \frac{1}{m}\right) \cdot R(\mathcal{O}_{\boldsymbol{p}})$ is not straightforward, even given the lower bounds of Lemma 5.8 and Lemma 5.9; indeed, $R(\mathcal{O}_{\boldsymbol{p}})$ cannot easily be computed, for an optimal choice of $\boldsymbol{p}$. We rely in our comparison on the lower-bounds of Lemma 5.8 and Lemma 5.9 instead; note in particular that even for the same choice of n, m, s, i.e., for a specific code, these might provide distinct lower-bounds on the rate. As mentioned above, even then it is not immediately clear how to efficiently encode and decode $\mathcal{C}_{\text{pilot}}(n)$.

Next, we consider the case of $k > 1$ and $\log(k) = o(n)$. We know from Corollary 5.2 that if $\limsup \frac{L_{\min}}{nk} \leq 1$ then any family of $(L_{\min}, L_{\max})$-multistrand torn-paper codes will only achieve vanishing asymptotic rate; hence we assume $L_{\min} = \lceil a \log(nk) \rceil$ for some $a > 1$. The following theorem summarizes our main results regarding $(L_{\min}, L_{\max})$-multistrand torn-paper codes.

Theorem 5.19. *Take n, k such that $k > 1$, $\log(k) = o(n)$, and let $L_{\min} = \lceil a \log(nk) \rceil$, for $a > 1$. There exists a linear run-time (in the substrings length, i.e., nk) encoder-decoder pair for $(L_{\min}, L_{\max})$-multistrand torn-paper codes achieving $1 - \frac{1}{a} - o_{nk}(1)$ asymptotic rate.*

Proof. Theorem 5.19 is justified by a simple amendment of Construction A. We encode $\boldsymbol{x} \in \Sigma^{kKm}$ into $\{\!\{\boldsymbol{z}^{(j)} : j \in [k]\}\!\}$, where $\left|\boldsymbol{z}^{(j)}\right| = n$ for all $j \in [k]$, as follows. We modify $I \triangleq \lceil \log(k\lceil n/L_{\min} \rceil) \rceil$ (recall, also, $\alpha \triangleq \left\lceil \frac{f(n)}{f(n)-1}(I+1) \right\rceil$) and $L_{\min} = \lceil a \log(nk) \rceil$. We then denote $\boldsymbol{x} = \boldsymbol{x}^{(0)} \circ \boldsymbol{x}^{(1)} \circ \cdots \circ \boldsymbol{x}^{(k-1)}$, where $\left|\boldsymbol{x}^{(j)}\right| = Km$ for all $j \in [k]$, and apply Algorithm 3 to $(\boldsymbol{x}^{(j)})_{j \in [k]}$ in succession; observe that every operation requires only $\lceil n/L_{\min} \rceil$ distinct indices, and we utilize available indices in order throughout the k operations.

We observe that the proofs of Lemma 5.15 and Lemma 5.16 hold without change, hence this amendment encodes into an $(L_{\min}, L_{\max})$-multistrand torn-paper code, which we denote $\mathcal{C}_A(n,k) \in \mathcal{X}_{n,k}$. Finally, following the proof of Theorem 5.18 we have

$$\begin{aligned}\text{red}(\mathcal{C}_A(n,k)) &= k\big(K(L_{\min} - m(N)) + L_{\min} + (n \bmod L_{\min})\big) \\ &\leq \frac{nk}{a}\left(1 + (1+o(1))\left(\frac{f(nk)}{\log(nk)} + \frac{1}{f(nk)}\right)\right)\end{aligned}$$

As in Theorem 5.18, for $f(n) = (1+o(1))\sqrt{\log(nk)}$ we have

$$\text{red}(\mathcal{C}_A(n,k)) \leq \frac{nk}{a}\left(1 + \frac{2+o(1)}{\sqrt{\log(nk)}}\right).$$

Table 5.1: Construction P (Lemma 5.8): Specific Parameters $(m, s, R(\mathcal{C}_{\text{pilot}}(n)))$.

$L_{\min}\backslash n$	60	250	4000	60,000	400,000	6,000,000
10	2, 5, 0.379	n/a	n/a	n/a	n/a	n/a
50	15, 3, **0.856**	10, 5, **0.844**	5, 10, **0.798**	3, 16, 0.667	2, 25, 0.5	2, 25, 0.5
100	n/a	10, 10, **0.9**	10, 10, **0.9**	6, 16, 0.833	5, 20, 0.8	4, 25, 0.75
300	n/a	n/a	32, 9, **0.968**	25, 12, **0.96**	20, 15, **0.95**	15, 20, **0.933**
1000	n/a	n/a	125, 8, **0.992**	100, 10, **0.989**	64, 15, **0.984**	50, 20, **0.98**

(Bold-face indicates Lemma 5.8 provides highest lower-bound on rate.)

Table 5.2: Construction P (Lemma 5.9): Specific Parameters $(m, s, R(\mathcal{C}_{\text{pilot}}(n)))$.

$L_{\min}\backslash n$	60	250	4000	60,000	400,000	6,000,000
10	2, 5, **0.45**	n/a	n/a	n/a	n/a	n/a
50	15, 3, 0.778	10, 5, 0.81	5, 10, 0.76	5, 10, **0.76**	4, 12, **0.719**	3, 16, **0.646**
100	n/a	10, 10, 0.855	10, 10, 0.855	10, 10, **0.855**	8, 12, 0.839	6, 16, 0.807
300	n/a	n/a	25, 12, 0.92	25, 12, 0.92	25, 12, 0.92	20, 15, 0.918
1000	n/a	n/a	50, 20, 0.956	50, 20, 0.956	50, 20, 0.956	50, 20, 0.956

(Bold-face indicates Lemma 5.9 provides highest lower-bound on rate. Background pattern indicates that the choice of m, s is only guaranteed by Lemma 5.9.)

Table 5.3: Construction A (Theorem 5.18): Specific Parameters $(f, I, N, K, R(\mathcal{C}_A(n)))$.

$L_{\min}\backslash n$	60	250	4000	60,000	400,000	6,000,000
10	n/a	n/a	n/a	n/a	n/a	n/a
50	n/a	2, 2, 40, 4, 0.56	3, 4, 38, 79, 0.711	3, 6, 35, 1199, 0.659	4, 7, 34, 7999, 0.66	4, 9, 31, 119999, 0.6
100	n/a	2, 1, 92, 1, 0.32	3, 3, 89, 39, 0.839	3, 5, 86, 599, 0.829	4, 6, 85, 3999, **0.84**	4, 8, 82, 59999, **0.81**
300	n/a	n/a	3, 2, 291, 12, 0.843	3, 4, 288, 199, 0.925	4, 6, 9, 285, 1332, 0.939	4, 8, 282, 19999, 0.93
1000	n/a	n/a	3, 1, 992, 3, 0.721	3, 3, 989, 59, 0.942	4, 5, 986, 399, 0.976	4, 7, 984, 5999, 0.976

(Bold-face indicates Theorem 5.18 provides highest lower-bound on rate.)

From Lemma 5.1 we have $\log|\mathcal{X}_{n,k}| \geq (n - \log(k))k$, concluding the proof. □

Again, by Corollary 5.2 and Theorem 5.18 the rate of the construction is asymptotically optimal.

5.4 Error-Correcting Torn-Paper Codes

In this section, we extend the study of torn-paper codes to a noisy setup. We consider two models of noise. The first one assumes that the encoded string, before segmentation, suffers at most some t substitution errors. The second model corresponds to the case where some of the segments are deleted during segmentation.

5.4.1 Substitution-Correcting Torn-Paper Codes

For a string $\boldsymbol{x}$, its *t-error torn-paper ball*, denoted by $\mathcal{BT}_{L_{\min}}^{L_{\max}}(\boldsymbol{x};t)$, is defined as the set of all possible $(L_{\min}, L_{\max})$-segmentations after introducing at most t errors to $\boldsymbol{x}$, that is,

$$\mathcal{BT}_{L_{\min}}^{L_{\max}}(\boldsymbol{x};t) \triangleq \bigcup_{\boldsymbol{y}\in B_t(\boldsymbol{x})} \mathcal{T}_{L_{\min}}^{L_{\max}}(\boldsymbol{y}),$$

where $B_t(\boldsymbol{x}) = \{\boldsymbol{y} : d_H(\boldsymbol{x},\boldsymbol{y}) \leq t\}$ is the radius-t Hamming ball centered at $\boldsymbol{x}$. A code $\mathcal{C}$ is called a *t-error single-strand torn-paper code* if for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{C}$, $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$, it holds that

$$\mathcal{BT}_{L_{\min}}^{L_{\max}}(\boldsymbol{x}_1;t) \cap \mathcal{BT}_{L_{\min}}^{L_{\max}}(\boldsymbol{x}_2;t) = \emptyset.$$

Our goal in this section is to show how to adjust Construction A in order to produce t-error single-strand torn-paper codes. We first explain the main ideas of the required modifications. Let $\boldsymbol{z} = \mathrm{Enc}_A(\boldsymbol{x}) \in \mathcal{C}_A(n)$ (encoded with Algorithm 3) and let $\mathcal{U} \in \mathcal{BT}_{L_{\min}}^{L_{\max}}(\boldsymbol{z};t)$ be an $(L_{\min}, L_{\max})$-segmentation of some word $\boldsymbol{z}'$, where $d_H(\boldsymbol{z},\boldsymbol{z}') \leq t$. The main task of the noiseless decoder of $\mathcal{C}_A(n)$ was to first calculate the index, and thus the location in $\boldsymbol{z}$, of every segment $\boldsymbol{u} \in \mathcal{U}$. However, in the presence of errors, calculating the index of a segment $\boldsymbol{u} \in \mathcal{U}$ based on the first (perhaps partial) occurrence of an encoded index within $\boldsymbol{u}$ might result with the misplacement of all the (perhaps partial) information blocks $\boldsymbol{y}_i$ that are contained in $\boldsymbol{u}$. Hence, a more careful approach is necessary for index decoding.

Before presenting our construction for t-error single-strand torn-paper codes, we introduce several additional required definitions. For a string $\boldsymbol{u}$, define $\mathcal{T}_{L_{\min}}^{+}(\boldsymbol{u})$ to be the multiset of non-overlapping $L_{\min}$-segments of $\boldsymbol{u}$, where the last segment is of length ℓ, $L_{\min} \leq \ell < 2L_{\min}$. A segment $\boldsymbol{w} \in \mathcal{T}_{L_{\min}}^{+}(\boldsymbol{u})$ is called *A-decodable* if, informally, Algorithm 4 returns (perhaps erroneous) output when given $\boldsymbol{w}$ as input. More formally, if $\boldsymbol{w}$ satisfies one of the following conditions.

1) $\boldsymbol{w}$ either contains a unique complete occurrence of $10^{f(n)}1$, or it doesn't contain complete occurrences but contains a cyclic occurrence (if $L_{\min} < |\boldsymbol{w}| < 2L_{\min}$, require instead that either the $L_{\min}$-prefix or the $L_{\min}$-suffix of $\boldsymbol{w}$ contain a cyclic occurrence).

2) $\boldsymbol{w}$ contains precisely two complete occurrences of $10^{f(n)}1$, and there exist a unique pair of occurrences (either complete or complete-to-suffix/prefix) whose locations are at distance precisely $L_{\min}$. Recall that $\boldsymbol{w}$ cannot contain more than two complete occurrences of $10^{f(n)}1$, except in the presence of errors, hence those cases can safely be discarded (see item 1 in the proof of Theorem 5.20 for a formal proof).

Let $\boldsymbol{w}$ be an A-decodable segment. Then, by definition, there is at least one occurrence (perhaps cyclic) of $10^{f(n)}1$ within $\boldsymbol{w}$ and, if there is more than a single occurrence, then there is exactly one pair of occurrences such that the difference between their locations is $L_{\min}$. Consider the α-segments of $\boldsymbol{w}$ preceding these occurrences as encoded indices; if the (first) occurrence of $10^{f(n)}1$ in $\boldsymbol{w}$ is at location $\ell < \alpha$, concatenate the $(\alpha-\ell)$-segment of $\boldsymbol{w}$ at location $L_{\min}+\ell-\alpha$, to the ℓ-prefix of $\boldsymbol{w}$, and consider the resulting length-α string to be a *cyclic encoded index*.

An A-decodable segment $\boldsymbol{w}$ is called *valid* if, informally, there appears at least one 'valid' encoded index in $\boldsymbol{w}$, and no conflicting pair of such indices. More formally, a valid segment $\boldsymbol{w}$ is an A-decodable segment that satisfies one of the following conditions:

1) $\boldsymbol{w}$ contains no complete encoded index, hence it contains only a cyclic encoded index.
2) $\boldsymbol{w}$ contains a single complete encoded index, and its parity symbol is correct.
3) $\boldsymbol{w}$ contains two complete encoded indices, and either exactly one of their parity symbols is correct, or both are correct and the indices are consecutive in the applied Gray code.

Construction B. *We construct a concatenated code, using Construction A as inner-code, and an arbitrary* $(K, q^{m(N)M}, 2t+1)_{q^{m(N)}}$ *error-correcting code* $\mathcal{C}_{\mathrm{EC}}$*, with an encoding algorithm*

$$\mathrm{Enc}_{\mathrm{EC}}\colon (\Sigma^{m(N)})^M \to (\Sigma^{m(N)})^K,$$

as outer-code (here, K, N *are the parameters of Construction A). The resulting* t*-error* $(L_{\min}, L_{\max})$*-single-strand torn-paper code is denoted* $\mathcal{C}_B(n)$*, with the associated encoder* $\mathrm{Enc}_B : \Sigma^{m(N)M} \to \Sigma^n$.

We observe the following property of Construction B. Assume one retrieves a noisy version $\boldsymbol{z}'$ of $\boldsymbol{z} = \mathrm{Enc}_B(\boldsymbol{x})$, e.g., from any reconstruction algorithm; further assume that $\boldsymbol{z}, \boldsymbol{z}'$ agree on all locations containing encoded indices $\boldsymbol{c}_i''$ or markers $10^{f(n)}1$ (as their locations in $\boldsymbol{z}$ are known a priori and do not depend on the information $\boldsymbol{x}$). Thus, one extracts from $\boldsymbol{z}'$ (perhaps erroneous) encoded information blocks, denoted $\boldsymbol{y}_i'$. Denote by e the number of encoded information blocks that were not recovered (e.g., due to conflicts in the reconstruction algorithm), and by s the number of encoded blocks that were recovered incorrectly (i.e., $\boldsymbol{y}_i' \neq \boldsymbol{y}_i$). Since the information string $(\boldsymbol{x}_i)_{i\in[M]} \in \Sigma^{m(N)M}$ is encoded using a $(K, q^{m(N)M}, 2t+1)_{q^{m(N)}}$ error-correcting code, it suffices that $2s + e \leq 2t$ to guarantee correct decoding.

In order to reconstruct a noisy version $\boldsymbol{z}'$ of $\boldsymbol{z}$, we define a modification of Algorithm 4, as follows. First, given an $(L_{\min}, L_{\max})$-segmentation $\mathcal{U} \in \mathcal{BT}_{L_{\min}}^{L_{\max}}(\boldsymbol{z}; t)$ we apply the reconstruction algorithm not directly to $\mathcal{U}$, but rather to valid segments in

$$\mathcal{T}_{L_{\min}}^{+}(\mathcal{U}) \triangleq \{\!\{\mathcal{T}_{L_{\min}}^{+}(\boldsymbol{u}) : \boldsymbol{u} \in \mathcal{U}\}\!\}.$$

Secondly, in case a valid $\boldsymbol{w} \in \mathcal{T}_{L_{\min}}^{+}(\mathcal{U})$ contains multiple (perhaps cyclic) occurrences of an encoded index, the algorithm selects one to decode by prioritizing complete occurrences over cyclic ones, and in the case of complete occurrences, accepting the first containing a correct parity symbol (since $\boldsymbol{w}$ is valid, such occurrence exists in this case). Decoding of the selected encoded index is then performed as described in Algorithm 4, and denoted by $\mathrm{Ind}'(\boldsymbol{w})$.

For an $(L_{\min}, L_{\max})$-segmentation $\mathcal{U} \in \mathcal{BT}_{L_{\min}}^{L_{\max}}(\boldsymbol{z}; t)$, we define the set

$$\mathcal{Z}(\mathcal{U}) \triangleq \left\{(\mathrm{Ind}'(\boldsymbol{w}), \boldsymbol{w}) : \boldsymbol{w} \in \mathcal{T}_{L_{\min}}^{+}(\mathcal{U}) \text{ is valid}\right\}.$$

If $(j, \boldsymbol{w}), (j, \boldsymbol{w}') \in \mathcal{Z}(\mathcal{U})$ for some j and $\boldsymbol{w} \neq \boldsymbol{w}'$, we define a restriction $\mathcal{Z}'(\mathcal{U})$ of $\mathcal{Z}(\mathcal{U})$ by including only the shortest, lexicographically-least, segment (i.e., $\mathcal{Z}'(\mathcal{U})$ defines a proper function).

Given the set $\mathcal{Z}'(\mathcal{U})$ we decode a string $\boldsymbol{z}'$ as follows.

1) Fill the encoded indices and the markers in $\boldsymbol{z}'$ in the correct locations as defined in Algorithm 3 (note again that these locations do not depend on the information).

2) Next, we iterate over any pair $(\mathrm{Ind}'(\boldsymbol{w}), \boldsymbol{w}) \in \mathcal{Z}'(\mathcal{U})$ and update $\boldsymbol{z}'$ with the symbols of the encoded blocks $\boldsymbol{y}_i$'s within $\boldsymbol{w}$; If there is a collision of symbols in the same position within an encoded block $\boldsymbol{y}'_i$ for some i, $i \in [K]$, we erase $\boldsymbol{y}'_i$ completely from $\boldsymbol{z}'$.

3) If an encoded block $\boldsymbol{y}'_i$ is partially filled at the end of the process (i.e., there are missing symbols within $\boldsymbol{y}'_i$) we erase the encoded block $\boldsymbol{y}'_i$.

The output $\boldsymbol{z}'$ of this decoding procedure over the segmentation $\mathcal{U} \in \mathcal{BT}^{L_{\max}}_{L_{\min}}(\boldsymbol{z}; t)$ is denoted by $\mathrm{Dec}_B(\mathcal{U}) \triangleq \boldsymbol{z}'$.

We now prove that $\mathcal{C}_B(n)$ is a t-error single-strand torn-paper code.

Theorem 5.20. *Let $\boldsymbol{z} = \mathrm{Enc}_B(\boldsymbol{x})$, $\mathcal{U} \in \mathcal{BT}^{L_{\max}}_{L_{\min}}(\boldsymbol{z}; t)$, and let $\boldsymbol{z}' = \mathrm{Dec}_B(\mathcal{U})$ be the noisy version of $\boldsymbol{z}$ reconstructed by the aforementioned algorithm. Then, it holds that $2s + e \leq 2t$, where e, s are defined as previously explained; i.e., any inner-channel error propagates as, at most, either one outer-channel error or two outer-channel erasures.*

Proof. By definition, $\mathcal{U}$ is obtained by first introducing up to t errors to $\boldsymbol{z}$, and then performing an $(L_{\min}, L_{\max})$-segmentation to the obtained word. For the rest of the proof, we fix an arbitrary $(L_{\min}, L_{\max})$-segmentation pattern, and for $\boldsymbol{z} \in \Sigma^n$ we denote $\mathcal{U} \in \mathcal{T}^{L_{\max}}_{L_{\min}}(\boldsymbol{z})$ obtained from this pattern by $\mathcal{U} = T(\boldsymbol{z})$. In particular, observe for $\|\boldsymbol{v}\| \leq t$ that $T(\boldsymbol{z} + \boldsymbol{v}) \in \mathcal{BT}^{L_{\max}}_{L_{\min}}(\boldsymbol{z}; t)$.

For convenience, we denote by $\boldsymbol{z}'_{\boldsymbol{v}} \triangleq \mathrm{Dec}_B(T(\boldsymbol{z} + \boldsymbol{v}))$, and by $e_{\boldsymbol{v}}$ (respectively, $s_{\boldsymbol{v}}$) the number of encoded information blocks $\boldsymbol{y}'_i$ in $\boldsymbol{z}'_{\boldsymbol{v}}$ which were not recovered (recovered erroneously). We shall prove the following proposition, which justifies the claim. Let $\boldsymbol{z} \triangleq \mathrm{Enc}_B(\boldsymbol{x})$, $\boldsymbol{v} \in \Sigma^n$ such that $\|\boldsymbol{v}\| \leq t$. Then $e_{\boldsymbol{v}} + 2s_{\boldsymbol{v}} \leq 2\|\boldsymbol{v}\|$.

The proof is done by induction on $\|\boldsymbol{v}\|$. First observe by Lemma 5.16 that $e_{\boldsymbol{0}} = s_{\boldsymbol{0}} = 0$ (here, $\boldsymbol{0}$ is the all-zero string). For the induction step, assume that the claim holds for any $\boldsymbol{v}' \in \Sigma^n$, $\|\boldsymbol{v}'\| < t$. Let $\boldsymbol{v} \in \Sigma^n$, $\|\boldsymbol{v}\| = t$. Take any $\boldsymbol{u}' \in \mathcal{T}^{+}_{L_{\min}}(T(\boldsymbol{z} + \boldsymbol{v}))$ affected by $t' > 0$ errors. Decompose $\boldsymbol{v} = \boldsymbol{v}' + \boldsymbol{v}''$ such that $\|\boldsymbol{v}'\| = t'$, $\|\boldsymbol{v}''\| = t - t'$, and $\boldsymbol{u}'$ contains the support of $\boldsymbol{v}'$. Consider the decoder output $\boldsymbol{z}'_{\boldsymbol{v}''}$; by the induction assumption,

$$e_{\boldsymbol{v}''} + 2s_{\boldsymbol{v}''} \leq 2(t - t').$$

We denote by $\boldsymbol{u}$ the segment corresponding to $\boldsymbol{u}'$ in $\boldsymbol{z}'_{\boldsymbol{v}''}$. Note that $\boldsymbol{u}$ contains no errors and its index is recovered correctly by the decoder. Hence, each encoded block that intersects $\boldsymbol{u}$ is either correct in $\boldsymbol{z}'_{\boldsymbol{v}''}$, or it is erased due to errors in other segments.

Denoting by δ the number of encoded information blocks intersecting $\boldsymbol{u}$, we let

(i) ρ_1 be the number of those recovered correctly in $\boldsymbol{z}'_{\boldsymbol{v}''}$;

(ii) ρ_2 be the number of those erased in $\boldsymbol{z}'_{\boldsymbol{v}''}$ due to to collisions resulting from incorrect index-decoding in other segments; and

(iii) ρ_3 be the number of those erased in $\boldsymbol{z}'_{\boldsymbol{v}''}$ due to erasures of other, intersecting, segments.

Observe that $\delta = \rho_1 + \rho_2 + \rho_3 \in \{1, 2, 3\}$, depending on $|\boldsymbol{u}|$ and its location). The rest of the proof is done by cases.

1) If $\boldsymbol{u}'$ is not valid, then all encoded information blocks intersecting $\boldsymbol{u}'$ are erased at the decoder. Hence, the ρ_1 correctly recovered blocks in $\boldsymbol{z}'_{v''}$ which intersect $\boldsymbol{u}$ are erased in $\boldsymbol{z}'_v$. In addition, each of the ρ_2 blocks corresponding to blocks intersecting $\boldsymbol{u}$ which are erased due to collisions, might instead cause incorrect recovery of encoded information blocks in $\boldsymbol{u}'$. Hence, $e_v \leq e_{v''} + \rho_1 - \rho_2$ and $s_v \leq s_{v''} + \rho_2$, and we note

$$\begin{aligned} e_v + 2s_v &\leq (e_{v''} + 2s_{v''}) + \rho_1 + \rho_2 \\ &\leq 2(t - t') + \delta = 2\|u\| - (2t' - \delta). \end{aligned}$$

   Since $t' \geq 1$, we have $e_v + 2s_v \leq 2\|u\|$ unless $\delta = 3$; however, in that case $\boldsymbol{u}$ contains two complete instances of $10^{f(n)}1$ whose locations are at distance $L_{\min}$, both preceded by complete occurrences of encoded indices, and since $\boldsymbol{u}'$ is not valid we have $t' > 1$, which also concludes the proof.

2) If $\boldsymbol{u}'$ is valid but its index is incorrectly decoded, then the ρ_1 encoded information blocks that are recovered correctly in $\boldsymbol{z}'_{v''}$ are erased in $\boldsymbol{z}'_v$, and ρ_2 encoded information blocks, corresponding to those intersecting $\boldsymbol{u}$ which are erased in $\boldsymbol{z}'_{v''}$ due to collisions, might be recovered incorrectly in $\boldsymbol{z}'_v$.

   Furthermore, the placement of $\boldsymbol{u}'$ at an incorrectly decoded location causes δ additional encoded information blocks to be either erased (due to collisions) or incorrectly recovered (where the correct blocks appear in invalid segments, i.e., are erased in $\boldsymbol{z}'_{v''}$). Denoting the number of blocks of the former type by δ_1, and the latter δ_2, we then have $e_v = e_{v''} + \rho_1 - \rho_2 + \delta_1 - \delta_2$ and $s_v \leq s_{v''} + \rho_2 + \delta_2$. Hence,

$$\begin{aligned} e_v + 2s_v &\leq (e_{v''} + 2s_{v''}) + \rho_1 + \rho_2 + \delta_1 + \delta_2 \\ &\leq 2(t - t') + 2\delta = 2\|v\| - 2(t' - \delta). \end{aligned}$$

   To conclude, we require $t' \geq \delta$. Indeed, observe that if $\delta = 2$ then $\boldsymbol{u}$ contains a complete occurrence of an encoded index followed by $10^{f(n)}1$, requiring $t' \geq 2$ for incorrect recovery. Likewise, if $\delta = 3$ then $\boldsymbol{u}$ contains two complete occurrences of encoded indices whose locations are at distance $L_{\min}$, each followed by $10^{f(n)}1$; incorrect recovery of the index therefore requires at least two errors in one of them in addition to further errors in the other index or $10^{f(n)}1$ marker, or an error in each $10^{f(n)}1$ marker in addition to further errors to generate such a marker at an alternative location, hence $t' \geq 3$ as well.

3) Finally, if the index of $\boldsymbol{u}'$ is decoded correctly (and, in particular, $\boldsymbol{u}'$ is valid), then recalling that the index of $\boldsymbol{u}$ is also decoded correctly, we clearly have $e_v = e_{v''}$. Since any error in $\boldsymbol{u}'$ can cause an error in at most a single encoded information block, we have that $s_v \leq s_{v''} + t'$. Hence,

$$e_v + 2s_v \leq e_{v''} + 2(s_{v''} + t') = (e_{v''} + 2s_{v''}) + 2t' \leq 2t = 2\|v\|.$$

□

Theorem 5.21. *Denote the redundancy of the outer-code $\mathcal{C}_{EC}$ used in Construction B by $\rho_{EC} \triangleq K - M$. Then, operating* Enc_A *as in Theorem 5.18, with $f(n) = (1+o(1))\sqrt{\log(n)}$, we have*

$$\begin{aligned}\mathrm{red}(\mathcal{C}_B(n)) &\leq \frac{n}{a}\left(1 + \frac{f(n)}{\log(n)} + \frac{1}{f(n)-1} + \frac{9+2/(f(n)-1)}{\log(n)} + \frac{4a}{q^{f(n)}} + \frac{2a^2+2}{n}\right) \\ &\quad + \rho_{EC}\left((a-1)\log(n) - 2\sqrt{\log(n)} - 11 - \frac{3}{\sqrt{\log(n)}-1} - \frac{4a\log(n)}{q^{\sqrt{\log(n)}}}\right) \\ &= \frac{n}{a}\left(1 + \frac{2+o(1)}{\sqrt{\log(n)}}\right) + \rho_{EC}\left((a-1)\log(n) - (2+o(1))\sqrt{\log(n)}\right).\end{aligned}$$

Furthermore, when $a > 2$ then the outer-code $\mathcal{C}_{EC}$ can be an MDS code and hence $\rho_{EC} = 2t$.

Proof. By Construction B, $\mathrm{red}(\mathcal{C}_B(n)) = \mathrm{red}(\mathcal{C}_A(n)) + \rho_{EC} \cdot m(N)$.

We recall from the proof of Theorem 5.18 that for $f(n) = \lceil \sqrt{\log(n)} \rceil$ it holds that

$$\begin{aligned}m(N) &\geq L_{\min} - \log(n) - f(n) - \frac{\log(n)}{f(n)-1} - 9 - \frac{2}{f(n)-1} - \frac{4a\log(n)}{q^{f(n)}} \\ &\geq (a-1)\log(n) - 2\sqrt{\log(n)} - 11 - \frac{3}{\sqrt{\log(n)}-1} - \frac{4a\log(n)}{q^{\sqrt{\log(n)}}},\end{aligned}$$

satisfying the former part of the claim.

Next, for $a > 2$ we observe that $m(N) > \log(n) - \log\log(n) + O_n(1) = \log(K)$, implying that an RS code may be used in Construction B, satisfying the latter part. □

Before concluding the section, we outline an extension of Construction B to the case $k > 1$, i.e., to *t-error multi-strand torn-paper codes*.

Corollary 5.22. *Take n, k such that $k > 1$, $\log(k) = o(n)$; let $L_{\min} = \lceil a\log(nk) \rceil$, for $a > 1$, and take some $L_{\max} \geq L_{\min}$. Amend Construction B as was done in Theorem 5.19 to Construction A, using a $(kK, q^{m(N)M}, 2t+1)_{q^{m(N)}}$ error-correcting code $\mathcal{C}_{\mathrm{EC}}$, with redundancy $\rho_{EC} \triangleq kK - M$. Then the resulting code $\mathcal{C}_B(n,k)$ is a t-error $(L_{\min}, L_{\max})$-multistrand torn-paper code, satisfying*

$$\mathrm{red}(\mathcal{C}_B(n,k)) \leq \frac{nk}{a}\left(1 + \frac{2+o(1)}{\sqrt{\log(nk)}}\right) + \rho_{EC}\left((a-1)\log(nk) - (2+o(1))\sqrt{\log(nk)}\right).$$

Proof. The proof of Theorem 5.20 applies without change. As in Theorem 5.19, we have

$$m(N) = (a-1)\log(nk) - (1+o(1))\left(f(nk) + \frac{\log(nk)}{f(nk)}\right),$$

and following the steps of Theorem 5.21, we have the claimed upper bound on redundancy, for $f(n) = (1+o(1))\sqrt{\log(nk)}$. □

5.4.2 Deletion-Correcting Torn-Paper Codes

For a string $\boldsymbol{x}$, its *t-deletion torn-paper ball*, $\mathcal{DT}^{L_{\max}}_{L_{\min}}(\boldsymbol{x};t)$, is defined as all the subsets with at most t missing segments of all the possible $(L_{\min}, L_{\max})$-segmentations of $\boldsymbol{x}$, that is,

$$\mathcal{DT}^{L_{\max}}_{L_{\min}}(\boldsymbol{x};t) \triangleq \bigcup_{S\in\mathcal{T}^{L_{\max}}_{L_{\min}}(\boldsymbol{x})} \left\{ S' \subseteq S : |S| - \left|S'\right| \leq t \right\}.$$

A code $\mathcal{C}$ is called a *t-deletion torn-paper code* if for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{C}$ it holds that $\mathcal{DT}^{L_{\max}}_{L_{\min}}(\boldsymbol{x}_1;t) \cap \mathcal{DT}^{L_{\max}}_{L_{\min}}(\boldsymbol{x}_2;t) = \emptyset$.

In this section, we utilize *burst-erasure-correcting (BEC) codes* in our constructions, which are defined next. For a string $\boldsymbol{x}$, its *t-burst L-erasures ball*, denoted by $\mathcal{B}^{L}_{\mathrm{BE}}(\boldsymbol{x};t)$, is defined as the set of all strings that can be obtained from $\boldsymbol{x}$ by at most t burst of erasures, each of length at most L. A code $\mathcal{C}$ is called a *t-burst L-erasure correcting code* if for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{C}$, $\mathcal{B}^{L}_{\mathrm{BE}}(\boldsymbol{x}_1;t) \cap \mathcal{B}^{L}_{\mathrm{BE}}(\boldsymbol{x}_2;t) = \emptyset$.

Next, we present a generic construction of t-deletion torn-paper codes. Let $\widehat{L}_{\max} \triangleq L_{\max} - \left\lceil \frac{L_{\max}}{L_{\min}} \right\rceil (\alpha + f(n) + 2)$. This construction is based on Construction A and assumes the existence of a systematic linear t-burst $\widehat{L}_{\max}$-erasure correcting code, denoted by $\mathcal{C}_{\mathrm{BEC}}$.

Construction C. *Let $\rho > 0$ be an integer that is determined next. This construction uses the following family of codes:*

Systematic BEC encoding. *Let* $\mathrm{Enc}_{\mathrm{BEC}} \colon \Sigma^{(K-\rho)N} \to \Sigma^{\rho_{\mathrm{BEC}}}$ *denote the systematic encoder of the code $\mathcal{C}_{\mathrm{BEC}}$, such that for any string $\boldsymbol{v} \in \Sigma^{(K-\rho)N}$, $\boldsymbol{v} \circ \mathrm{Enc}_{\mathrm{BEC}}(\boldsymbol{v}) \in \mathcal{C}_{\mathrm{BEC}}$ (for convenience we assume that $\mathrm{Enc}_{\mathrm{BEC}}(\boldsymbol{v})$ returns only the encoded systematic redundancy symbols). The redundancy of this encoder is denoted by ρ_{BEC}. The parameter ρ is defined $\rho \triangleq \left\lceil \frac{1}{N} \left\lfloor \rho_{\mathrm{BEC}} \cdot \frac{f(n)}{f(n)-1} \right\rfloor \right\rceil$.*

Next, we utilize a generalized concatenated coding approach, where Construction A is used as inner-code for $K-\rho$ information blocks, and with a slight adjustment also for the ρ redundant blocks, as follows:

1) The length of the input string $\boldsymbol{x}$. *The input of this construction is $\boldsymbol{x} \in \Sigma^{(K-\rho)m(N)}$. That is, this construction has additional redundancy of $\rho m(N)$ symbols compared to Construction A. The input string is divided to $K-\rho$ information blocks each of length $m(N)$, denoted by $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{K-\rho-1}$.*

2) The generation of the encoded blocks $\boldsymbol{y}_i$'s. *The first $K-\rho$ blocks are generated from the corresponding $\boldsymbol{x}_i$'s using the RLL encoder E^{RLL}_m similarly to Construction A. Let $\boldsymbol{y}^* \triangleq \boldsymbol{y}_0 \circ \cdots \circ \boldsymbol{y}_{K-\rho-1} \in \Sigma^{(K-\rho)N}$ denote their concatenation. Next, we apply* $\mathrm{Enc}_{\mathrm{BEC}}$ *to obtain $\boldsymbol{w} \triangleq \mathrm{Enc}_{\mathrm{BEC}}(\boldsymbol{y}^*)$, and denote by $\boldsymbol{w}^*$ the result of inserting '1's into $\boldsymbol{w}$ at every location divisible by $f(n)$ (in particular, $\boldsymbol{y}^* \circ \boldsymbol{w}^*$ does not contain a length-$f(n)$ zero-run). Then, $\boldsymbol{w}^*$ is divided to the remaining segments $\boldsymbol{y}_{K-\rho'}, \ldots, \boldsymbol{y}_{K-1} \in \Sigma^N$ (if $|\boldsymbol{w}^*|$ is not a multiple of N, $\boldsymbol{y}_{K-1}$ is padded with 1's to length N). Note that the parameter ρ satisfies $\rho N \geq \left\lfloor \rho_{\mathrm{BEC}} \cdot \frac{f(n)}{f(n)-1} \right\rfloor = |\boldsymbol{w}^*|$, hence one may continue to follow the steps of Construction A without change.*

We now indeed continue identically to Construction A. That is, an index and a marker are appended to the beginning of each encoded block $\boldsymbol{y}_i$ to construct a segment $\boldsymbol{z}_i$ of length $L_{\min}$. Then, $\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{K-1}$ are concatenated along with $\boldsymbol{z}_K \circ 0^{n \bmod L_{\min}} = \boldsymbol{c}''_K \circ 10^{f(n)} 10^{N+(n \bmod L_{\min})}$ to obtain the encoded output string $\boldsymbol{z} \in \Sigma^n$.

Let $\mathcal{C}_{\mathrm{del}}(n)$ denote the constructed code. The correctness of Construction C and redundancy calculation are proved in the next theorem.

Theorem 5.23. *$\mathcal{C}_{\mathrm{del}}(n)$ is a t-deletion torn-paper code. Furthermore, it holds that*

$$\mathrm{red}(\mathcal{C}_{\mathrm{del}}(n)) = \mathrm{red}(\mathcal{C}_A(n)) + m(N) \left\lceil \frac{1}{N} \left\lfloor \rho_{\mathrm{BEC}} \cdot \frac{f(n)}{f(n)-1} \right\rfloor \right\rceil.$$

Proof. Let $\boldsymbol{z} \in \mathcal{C}_{\mathrm{del}}(n)$ be the encoded codeword of the input string $\boldsymbol{x}$, and take $\mathcal{U} \in \mathcal{DT}^{L_{\max}}_{L_{\min}}(\boldsymbol{z}; t)$. We shall prove that one can uniquely decode $\boldsymbol{x}$.

From Lemma 5.16, for every $\boldsymbol{u} \in \mathcal{U}$ which is not a substring of the suffix of length $(n \mod L_{\min}) + N + f(n)$ of $\boldsymbol{z}$, its index $\mathrm{Ind}(\boldsymbol{u})$ can be decoded using Algorithm 4. The string $\boldsymbol{z}$ can then be reconstructed by the locations of each received segment, with some segments erased (at identifiable locations). Let $\boldsymbol{z}' \in (\Sigma \cup \{?\})^n$ denote this partially reconstructed string, where '?' stands for erased symbols.

From the definition of $\mathcal{DT}^{L_{\max}}_{L_{\min}}(\boldsymbol{z}; t)$, at most t segments of $\boldsymbol{z}$ are missing from $\mathcal{U}$. Therefore, $\boldsymbol{z}' \in \mathcal{B}^{L_{\max}}_{\mathrm{BE}}(\boldsymbol{z}; t)$. By removing coordinates of $\boldsymbol{z}'$ corresponding to indices or markers (including '?' symbols), a string $\boldsymbol{y}' \in \mathcal{B}^{\widehat{L}_{\max}}_{\mathrm{BE}}\left(\boldsymbol{y}_0 \circ \cdots \circ \boldsymbol{y}_{K-1}; t\right)$ is obtained, since there are at most $L_{\max} - \widehat{L}_{\max} = \left\lceil \frac{L_{\max}}{L_{\min}} \right\rceil (\alpha + f(n) + 2)$ symbols in any $L_{\max}$-segment of $\boldsymbol{z}$ belonging to either index or marker.

Finally, we remove from $\boldsymbol{y}'$ coordinates corresponding to '1's inserted into $\boldsymbol{w}^*$; thus, we obtain a string $\widehat{\boldsymbol{y}} \in \mathcal{B}^{\widehat{L}_{\max}}_{\mathrm{BE}}(\boldsymbol{y}^* \circ \boldsymbol{w}; t)$. A decoder for $\mathcal{C}_{\mathrm{BEC}}$ may be invoked on $\widehat{\boldsymbol{y}}$ to retrieve $\boldsymbol{y}^* = \boldsymbol{y}_0 \circ \cdots \circ \boldsymbol{y}_{K-\rho-1}$, and consequently $\boldsymbol{x}$ is obtained by applying the RLL decoder to each $\boldsymbol{y}_i$, $i \in [K-\rho]$.

To conclude the proof we observe that the asserted redundancy follows by definition, as precisely $\rho m(N)$ less information symbols are input at the encoder, in comparison to Construction A. □

Next, we note that an extension to the case $k > 1$, i.e., to *t-deletion multi-strand torn-paper codes*, is again straightforward.

Corollary 5.24. *Amending Construction C, one constructs a t-deletion multi-strand torn-paper code $\mathcal{C}_{\mathrm{del}}(n,k)$ with redundancy*

$$\mathrm{red}(\mathcal{C}_{\mathrm{del}}(n,k)) = \mathrm{red}(\mathcal{C}_A(n,k)) + m(N) \left\lceil \frac{1}{N} \left\lfloor \rho_{\mathrm{BEC}} \cdot \frac{f(n)}{f(n)-1} \right\rfloor \right\rceil.$$

Proof. Here, an information string $\boldsymbol{x} \in \Sigma^{(kK-\rho)m(N)}$ is encoded with E^{RLL}_m into $\boldsymbol{y}^*$, and $\boldsymbol{w}^*$ is obtained utilizing a systematic BEC encoder on strings in $\Sigma^{(kK-\rho)N}$. It is segmented into $\boldsymbol{y}_{kK-\rho}, \ldots, \boldsymbol{y}_{kK} \in \Sigma^N$; again, observing $\rho N \geq \rho_{\mathrm{BEC}}$ assures that this is possible. Then, each

K segments $\boldsymbol{y}_j$ are encoded, in order, with the remaining steps of Algorithm 3, where again $I \triangleq \lceil \log(k\lceil n/L_{\min}\rceil)\rceil$ and $L_{\min} = \lceil a\log(nk)\rceil$, and indices are utilized by each operation in succession. It is straightforward that the proof of Theorem 5.23 can be followed to show that $\mathcal{C}_{\text{del}}(n,k)$ is a t-deletion multi-strand torn-paper code, with the above redundancy. □

Before concluding the section, we discuss the cases of $t \in \{1,2\}$, in which more is known on the construction of BEC codes.

For $t = 1$, we use a systematic interleaving parity BEC code as the code $\mathcal{C}_{\text{BEC}}$. Namely, the redundancy string $\boldsymbol{w} = \text{Enc}_{\text{BEC}}(\boldsymbol{y}^*)$ is of length $\rho_{\text{BEC}} = \widehat{L}_{\max}$, and

$$w_i \triangleq \sum\nolimits_{k\in\left[\left\lceil \frac{(K-\rho)N-i}{\widehat{L}_{\max}}\right\rceil\right]} y^*_{i+k\widehat{L}_{\max}}$$

for all $i \in [\widehat{L}_{\max}]$, i.e., w_i is a single parity symbol for $\left(y^*_i, y^*_{i+\widehat{L}_{\max}}, \ldots\right)$. Denote this code by $\mathcal{C}_{\text{del},1}$.

For $t = 2$, we state for completeness the following basic proposition which draws the connection between burst-error-correcting codes and burst-erasure-correcting codes. We note that this fact has been mentioned before in [7], for a single burst of errors.

Lemma 5.25. *For $0 < \ell \le n$ and $\boldsymbol{x}, \boldsymbol{y} \in \Sigma^n$, it holds that $\boldsymbol{x}, \boldsymbol{y}$ are confusable under t bursts of errors of lengths at most ℓ if and only if they are confusable under $2t$ bursts of erasures of lengths at most ℓ.*

Proof. Denote $\boldsymbol{x} = (x_j)_{j\in[n]}$, $\boldsymbol{y} = (y_j)_{j\in[n]}$, and $I_i \triangleq \bigcup_{j\in[t]}(k^{(i)}_j + [\ell])$, for $i = 0,1$ and some $\left\{k^{(i)}_j\right\}_{j\in[t]} \subseteq [n]$. Assume there exist $\boldsymbol{e}^{(0)}, \boldsymbol{e}^{(1)} \in \Sigma^n$ such that $\boldsymbol{x} + \boldsymbol{e}^{(0)} = \boldsymbol{y} + \boldsymbol{e}^{(1)}$, and $\text{supp}(\boldsymbol{e}^{(i)}) \subseteq I_i$, $i = 1,2$. Then, one observes that $\boldsymbol{x}_{[n]\setminus(I_0\cup I_1)} = \boldsymbol{y}_{[n]\setminus(I_0\cup I_1)}$.

Conversely, assume $\boldsymbol{x}_{[n]\setminus I} = \boldsymbol{y}_{[n]\setminus I}$, where $I \subseteq \bigcup_{j\in[2t]}(k_j + [\ell])$ for some $\left\{k_j\right\}_{j\in[2t]} \subseteq [n]$, and without loss of generality $\left\{k_j\right\}_{j\in[2t]}$ are increasing, and $k_j \le k_{t+1} - \ell$ for all $j \le t$. Let $I_i \triangleq \bigcup_{j\in(it+[t])}(k_j + [\ell])$ for $i = 1,2$, and observe that $I_0 \cup I_1 = I$, $I_0 \cap I_1 = \emptyset$. For $i = 1,2$ and $j \in [n]$, let

$$e^{(i)}_j \triangleq \begin{cases} (-1)^i(y_j - x_j), & j \in I_i; \\ 0, & \text{otherwise.} \end{cases}$$

Then, denoting $\boldsymbol{e}^{(i)} = (e^{(i)}_j)_{j\in[n]}$ for $i = 1,2$, we have $\boldsymbol{x} + \boldsymbol{e}^{(0)} = \boldsymbol{y} + \boldsymbol{e}^{(1)}$, which completes the proof. □

A construction of 2-deletion torn-paper codes is derived from Construction C, using a BEC code for $t = 2$. Hence, by Lemma 5.25 one may use an $\widehat{L}_{\max}$-burst error-correcting code. Observe that Construction C requires a systematic encoder, which is guaranteed by several prior works with redundancy at most $\log((K-\rho)N) + \widehat{L}_{\max}$; see, e.g. [1, 2]. These constructions require the alphabet Σ to be a field, and are linear and cyclic, which ensures the existence of a systematic encoder. For simplicity of derivation we bound this redundancy (from above) by $\widehat{L}_{\max} + \log(n)$. Let $\mathcal{C}_{\text{del},2}$ denote this code.

The next corollary summarizes these results. For convenience, denote the difference

$$\Delta\,\mathrm{red}(\mathcal{C}(n)) \triangleq \mathrm{red}(\mathcal{C}(n)) - \mathrm{red}(\mathcal{C}_A(n)),$$

for a t-deletion torn-paper code $\mathcal{C}(n) \subseteq \Sigma^n$.

Corollary 5.26. *For a prime power q and all admissible values of n and $f(n)$ in Construction A, where $f(n) = \omega(1)$, $f(n) = o(\log(n))$ and with the RLL encoders of [20, 37], it holds that*

$$\Delta\,\mathrm{red}(\mathcal{C}_{\mathrm{del},1}(n)) \leq \widehat{L}_{\max} \cdot \frac{f(n)}{f(n)-1},$$
$$\Delta\,\mathrm{red}(\mathcal{C}_{\mathrm{del},2}(n)) \leq (\widehat{L}_{\max} + \log(n)) \cdot \frac{f(n)}{f(n)-1}.$$

In particular, for $f(n) = (1+o(1))\sqrt{\log(n)}$,

$$\Delta\,\mathrm{red}(\mathcal{C}_{\mathrm{del},1}(n)) \leq \widehat{L}_{\max}\left(1 + \frac{1-o(1)}{\sqrt{\log(n)}}\right),$$
$$\Delta\,\mathrm{red}(\mathcal{C}_{\mathrm{del},2}(n)) \leq (\widehat{L}_{\max} + \log(n))\left(1 + \frac{1-o(1)}{\sqrt{\log(n)}}\right).$$

Note that if $L_{\max} = o(n)$ the asymptotic rate of $\mathcal{C}_{\mathrm{del},1}(n)$ and $\mathcal{C}_{\mathrm{del},2}(n)$ is asymptotically equal to the rate of $\mathcal{C}_A(n)$. Thus, efficient encoding and decoding of t-deletion torn-paper codes, $t = 1, 2$, is possible at rates arbitrarily close to the optimum.

5.5 Conclusion

In this paper, we study the adversarial torn-paper channel, for which we present fundamental bounds and code constructions. We further study several extensions of this model, including multi-strand storage, substitution errors, or incomplete coverage. Importantly, our proposed constructions have linear-run-time encoders and decoders, and the resulting codes achieve asymptotically optimal rates.

We mention again that the adversarial model we assume in this work is chosen to enable analysis in the worst-case. More realistically, an adversarial channel where the *average* of the received segments' lengths is bounded from below might be analyzed; unfortunately, this channel turns out to be hard to analyze in the worst-case, and such analysis is left for future works. It will be remarked that by the same methods of Lemma 5.1, it can be shown that the capacity of such an adversarial channel is bounded from above by $1 - \frac{1}{a}$, where the lower bound on the average segment length is chosen to be $a\log(n)$. Coding for this channel appears to be more challenging; we point to the fact that an adversary is able to segment a non-vanishing fraction of the channel input into short substrings as a likely reason for that difficulty.

A naive solution, where the lower bound on the average segment length is $a\log(n)$ and $a > \frac{q}{q-1}$, is to apply Construction C with parameter a' satisfying $1 < a' < (1 - \frac{1}{q})a$; the decoder then discards any received segment shorter than $a'\log(n)$, creating at most $\frac{n}{a\log(n)}$ bursts

of erasures of lengths at most $(a'-1)\log(n)$ (in the reconstructed information sequence). To recover the information, a BEC code $\mathcal{C}_{\text{BEC}}$ is used; since $\frac{1}{Km(N)}(a'-1)\log(n)\frac{n}{a\log(n)} = \frac{(a'-1)/a}{Km(N)/n} = \frac{a'}{a} + o(1) < 1 - \frac{1}{q}$, a positive-rate BEC code exists for all a' in the permissible range (since positive-rate $\frac{a'-1}{a}n$-erasure-correcting codes exist in $\Sigma^{Km(N)}$); hence a' can be optimized according to Theorem 5.23, i.e., to maximize the achieved rate of $(1+o_n(1))\left(1-\frac{1}{a'}\right)\cdot R(\mathcal{C}_{\text{BEC}})$. (Naively, one might utilize codes correcting $\frac{n}{a\log(n)}\cdot(a'-1)\log(n)$ erasures; by the GV bound, the achievable rate of this construction is at least $(1+o_n(1))\left(1-\frac{1}{a'}\right)\left(1-H_q(\frac{a'}{a})\right)$.) Alternatively, if any integer a' falls within the given range, Construction P can also be used with parameter a', where again segments shorter than $(a'+o(1))\log(n)$ are discarded at the decoder, and reconstructed based on a BEC code $\mathcal{C}_{\text{BEC}}$ correcting up to $\frac{n}{a\log(n)}$ bursts of erasures of lengths at most $(1+o(1))\log(n)$ in $\Sigma^{n/a'}$; however, the achieved rate of this construction is similarly $\left(1-\frac{1}{a'}\right)\cdot R(\mathcal{C}_{\text{BEC}})$, and applicable BEC codes are equivalent (i.e., they correct the same number of bursts, of length $(1+o(1))\log(n)$ in $\Sigma^{n/a'}$ instead of length $(a'-1)\log(n)$ in $\Sigma^{Km(N)} = \Sigma^{(1-1/a'+o(1))n}$).

Finally, for future research, we believe that applying our methods to a generalized channel, including multiple sources of noise concurrently, one may achieve similar results. Studying the channel under edit-errors, including insertions/deletions in addition to substitutions, is also of great interest for applications to DNA data storage.

Acknowledgments

The authors gratefully acknowledge the valuable insight and advice offered to us by the two anonymous reviewers and associate editor, which were instrumental in streamlining the presentation of this paper.

Bibliography

[1] K. A. S. Abdel-Ghaffar, "On the existence of optimum cyclic burst correcting codes over GF(q)," *IEEE Trans. on Inform. Theory*, vol. 34, no. 2, pp. 329–332, Mar. 1988.

[2] K. A. S. Abdel-Ghaffar, R. J. McEliece, A. M. Odlyzko, and H. C. A. van Tilborg, "On the existence of optimum cyclic burst-correcting codes," *IEEE Trans. on Inform. Theory*, vol. 32, no. 6, pp. 768–775, Nov. 1986.

[3] J. Acharya, H. Das, O. Milenkovic, A. Orlitsky, and S. Pan, "String reconstruction from substring compositions," *SIAM J. Discrete Math.*, vol. 29, no. 3, pp. 1340–1371, 2015.

[4] F. Balado, "Capacity of DNA data embedding under substitution mutations," *IEEE Trans. on Inform. Theory*, vol. 59, no. 2, pp. 928–941, Feb. 2013.

[5] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "A DNA-based archival storage system," *ACM SIGPLAN Notices*, vol. 51, no. 4, pp. 637–649, Mar. 2016.

[6] G. Bresler, M. Bresler, and D. Tse, "Optimal assembly for high throughput shotgun sequencing," *BMC Bioinformatics*, vol. 14, no. 5, p. S18, Jul. 2013.

[7] R. T. Chien, L. R. Bahl, and D. Tang, "Correction of two erasure bursts (corresp.)," *IEEE Trans. on Inform. Theory*, vol. 15, no. 1, pp. 186–187, Jan. 1969.

[8] C.-S. Chin, D. H. Alexander, P. Marks, A. A. Klammer, J. Drake, C. Heiner, A. Clum, A. Copeland, J. Huddleston, E. E. Eichler, S. W. Turner, and J. Korlach, "Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data," *Nature Methods*, vol. 10, no. 6, pp. 563–569, Jun. 2013.

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

[10] Contributing Members, "Preserving our digital legacy: an introduction to DNA data storage," The DNA Data Storage Alliance, White Paper, Jun. 2021. [Online]. Available: https://dnastoragealliance.org/dev/wp-content/uploads/2021/06/DNA-Data-Storage-Alliance-An-Introduction-to-DNA-Data-Storage.pdf

[11] N. G. de Bruijn and T. van Aardenne-Ehrenfest, "Circuits and trees in oriented linear graphs," *Simon Stevin*, vol. 28, pp. 203–217, 1951.

[12] O. Elishco, R. Gabrys, M. Médard, and E. Yaakobi, "Repeat-free codes," *IEEE Trans. on Inform. Theory*, vol. 67, no. 9, pp. 5749–5764, Sep. 2021.

[13] Y. Erlich and D. Zielinski, "DNA fountain enables a robust and efficient storage architecture," *Science*, vol. 355, no. 6328, pp. 950–954, Mar. 2017.

[14] R. Gabrys and O. Milenkovic, "Unique reconstruction of coded strings from multiset substring spectra," *IEEE Trans. on Inform. Theory*, vol. 65, no. 12, pp. 7682–7696, Dec. 2019.

[15] S. Ganguly, E. Mossel, and M. Racz, "Sequence assembly from corrupted shotgun reads," in *Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT), Barcelona, Spain*, Jul. 2016, pp. 265–269.

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

[17] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angewandte Chemie International Edition*, vol. 54, no. 8, pp. 2552–2555, 2015.

[18] R. Heckel, I. Shomorony, K. Ramchandran, and D. N. C. Tse, "Fundamental limits of DNA storage systems," in *Proceedings of the 2017 IEEE International Symposium on Information Theory (ISIT), Aachen, Germany*, Jun. 2017, pp. 3130–3134.

[19] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "An upper bound on the capacity of the DNA storage channel," in *Proceedings of the 2019 IEEE Information Theory Workshop (ITW), Visby, Sweden*, Aug. 2019.

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

[21] N. J. Loman, J. Quick, and J. T. Simpson, "A complete bacterial genome assembled de novo using only nanopore sequencing data," *Nature Methods*, vol. 12, no. 8, pp. 733–735, Aug. 2015.

[22] A. Motahari, K. Ramchandran, D. Tse, and N. Ma, "Optimal DNA shotgun sequencing: Noisy reads are as good as noiseless reads," in *Proceedings of the 2013 IEEE International Symposium on Information Theory (ISIT), Istanbul, Turkey*, Jul. 2013, pp. 1640–1644.

[23] A. S. Motahari, G. Bresler, and D. N. C. Tse, "Information theory of DNA shotgun sequencing," *IEEE Trans. on Inform. Theory*, vol. 59, no. 10, pp. 6273–6289, Oct. 2013.

[24] S. Nassirpour, I. Shomorony, and A. Vahid, "Reassembly codes for the chop-and-shuffle channel," *arXiv preprint arXiv:2201.03590*, 2022.

[25] L. Organick, S. D. Ang, Y.-J. Chen *et al.*, "Random access in large-scale DNA data storage," *Nature Biotechnology*, vol. 36, no. 3, pp. 242–248, Mar. 2018.

[26] A. N. Ravi, A. Vahid, and I. Shomorony, "Capacity of the torn paper channel with lost pieces," in *Proceedings of the 2021 IEEE International Symposium on Information Theory (ISIT), Melbourne, Victoria, Australia*, Jul. 2021, pp. 1937–1942.

[27] ——, "Coded shotgun sequencing," *IEEE Journal on Selected Areas in Information Theory*, vol. 3, no. 1, pp. 147–159, Mar. 2022.

[28] S. L. Salzberg, "Mind the gaps," *Nature Methods*, vol. 7, no. 2, pp. 105–106, Feb. 2010.

[29] I. Shomorony, T. Courtade, and D. Tse, "Do read errors matter for genome assembly?" in *Proceedings of the 2015 IEEE International Symposium on Information Theory (ISIT), Hong Kong, China*, Jun. 2015, pp. 919–923.

[30] I. Shomorony and R. Heckel, "Capacity results for the noisy shuffling channel," in *Proceedings of the 2019 IEEE International Symposium on Information Theory (ISIT), Paris, France*, Jul. 2019, pp. 762–766.

[31] I. Shomorony, G. M. Kamath, F. Xia, T. A. Courtade, and D. N. Tse, "Partial DNA assembly: A rate-distortion perspective," in *Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT), Barcelona, Spain*, Jul. 2016, pp. 1799–1803.

[32] I. Shomorony and A. Vahid, "Torn-paper coding," *IEEE Trans. on Inform. Theory*, vol. 67, no. 12, pp. 7904–7913, Dec. 2021.

[33] J. Spencer, "Asymptotic lower bounds for Ramsey functions," *Discrete Mathematics*, vol. 20, pp. 69–76, 1977.

[34] R. R. Varshamov and G. M. Tenengolts, "Code correcting single asymmetric errors (in Russian)," *Automatika i Telemkhanika*, vol. 26, no. 2, pp. 288–292, 1965.

[35] N. Weinberger and N. Merhav, "The DNA storage channel: Capacity and error probability bounds," *IEEE Trans. on Inform. Theory*, vol. 68, no. 9, pp. 5657–5700, Sep. 2022.

[36] P. C. Wong, K.-k. Wong, and H. Foote, "Organic data memory using the DNA approach," *Communications of the ACM*, vol. 46, no. 1, pp. 95–98, Jan. 2003.

[37] Y. Yehezkeally, D. Bar-Lev, S. Marcovich, and E. Yaakobi, "Generalized unique reconstruction from substrings," *IEEE Trans. on Inform. Theory*, 2023.

[38] Y. Yehezkeally and N. Polyanskii, "On codes for the noisy substring channel," in *Proceedings of the 2021 IEEE International Symposium on Information Theory (ISIT), Melbourne, Victoria, Australia*, Jul. 2021, pp. 1700–1705.

Chapter 6

Optimal Almost-Balanced Sequences

Daniella Bar-Lev, Adir Kobovich, Orian Leitersdorf, and Eitan Yaakobi

Abstract

This paper presents a novel approach to address the constrained coding challenge of generating *almost-balanced sequences*. While strictly balanced sequences have been well studied in the past, the problem of designing efficient algorithms with small redundancy, preferably constant or even a single bit, for almost balanced sequences has remained unsolved. A sequence is $\varepsilon(n)$-*almost balanced* if its Hamming weight is between $0.5n \pm \varepsilon(n)$. It is known that for any algorithm with a constant number of redundancy bits, $\varepsilon(n)$ has to be in the order of $\Theta(\sqrt{n})$. However, most prior solutions with a single redundancy bit required $\varepsilon(n)$ to be a linear shift from $n/2$. Employing an iterative method and arithmetic coding, our emphasis lies in constructing almost balanced codes with a single redundancy bit. Notably, our method achieves the *optimal* balanced order of $\Theta(\sqrt{n})$. Additionally, we extend our method to the non-binary case, considering q-ary almost polarity-balanced sequences for even q, and almost symbol-balanced for $q = 4$. Overall, our work advances the field of constrained coding by providing a novel method for handling almost-balanced sequences, for both, binary and non-binary alphabet.

6.1 Introduction

Constrained codes have a long history in information theory, with applications to data storage and transmission. In the broadest setting, raw data in such applications is encoded (in a one-to-one manner) into a set of words $\mathcal{S}$ over some alphabet Σ that satisfy prescribed rules. Some rules are imposed due to physical limitations, such as those dictated by energy compliance or by memory cell wear, and are typically translated into cost constraints. Others are imposed as a preventive measure to keep the storage device in a sufficiently-reliable operation region. A celebrated result in constrained coding theory by Knuth has analyzed strictly balanced binary sequences or sequences with a fixed weight [1]. We consider in this work the *almost-balanced*

constraint which generalizes the well-known balanced Knuth codes [1] by requiring that the entire message possess a Hamming weight of *approximately* $n/2$.

One motivating application of this work is DNA storage, where *almost balanced GC content* is necessary [2]. During the storage phase in DNA strands, media degradation, and in particular breaks, can arise in DNA due to factors that include radiation, humidity, and high temperatures. In [3], the authors proposed to encapsulate the stored DNA in a silica substrate and then to employ custom error-correcting codes to mitigate the effects of these errors. Another approach to dealing with media degradation is to generate strands of DNA that have approximately balanced GC-content, and this approach has been leveraged in several existing works such as [4–6].

The construction of efficient balanced codes has been extensively studied; see e.g. [1,7–10], and extensions to non-binary balanced codes have been considered in [11–15]. Codes that combine the balanced property with certain other constraints, such as run-length limitations, have also been addressed for example in [16]. However, the problem of almost balanced sequences with Hamming weight between $0.5n \pm \varepsilon(n)$ has received a little attention. Under this framework, the goal is to find the optimal number of redundant bits as a function of $\varepsilon(n)$, where $\varepsilon(n)$ can be a function of n, e.g. linear in n, $\log n$, or a constant. No less important is the design of such algorithms.

While Knuth's algorithm is an efficient scheme to strictly balance an arbitrary sequence with $\log n + o(\log n)$ redundancy bits, designing an efficient encoder and decoder with less redundancy or even only a single bit is a non-trivial task. The best-known approach for minimizing $\varepsilon(n)$ is enumerative coding [17], which can achieve the lower bound of $\varepsilon(n) = \Omega(\sqrt{n})$, however optimizing it for efficient memory and time usage requires sophisticated and cumbersome techniques (see e.g., [18]). In contrast, simpler solutions with linear time complexity result in $\varepsilon(n)$ that is linear with n [19]. In this work, we present an explicit encoder that uses a single redundancy bit to balance binary sequences for $\varepsilon(n) = \Theta(\sqrt{n})$. Additionally, we show that on average our algorithm performs $\mathcal{O}(n)$ multiplications of rational numbers.

The rest of this paper is organized as follows. In Section 6.2 we introduce the definitions that will be utilized throughout the paper and present the arithmetic coding method. Our construction for binary almost-balanced sequences is presented in Section 6.3 and generalizations for almost polarity-balanced and almost symbol-balanced for non-binary alphabet are presented in Section 6.4. Section 6.5 concludes this paper.

6.2 Definitions, Related Works, and Arithmetic Coding

6.2.1 Definitions

Let $\Sigma_q = \{0, 1, \ldots, q-1\}$ be the alphabet of size q and let Σ_q^n be the set of all sequences of length n over Σ_q. The Hamming weight of a sequence $\boldsymbol{x} \in \Sigma_q^n$, denoted by $w(\boldsymbol{x})$, is the number of non-zero symbols in $\boldsymbol{x}$. The concatenation of two sequences $\boldsymbol{x}$ and $\boldsymbol{y}$ is denoted by $\boldsymbol{x} \circ \boldsymbol{y}$.

A binary sequence $\boldsymbol{x} \in \Sigma_2^n$ is called *balanced* if the number of zeros and ones is identical, i.e., if $w(\boldsymbol{x}) = \frac{n}{2}$. We similarly define an *almost-balanced* binary sequence as follows.

Definition 6.1. *A sequence $\boldsymbol{x} \in \Sigma_2^n$ is called* $\varepsilon(n)$-almost balanced *if*

$$w(\boldsymbol{x}) \in \left[\frac{n}{2} - \varepsilon(n), \frac{n}{2} + \varepsilon(n)\right].$$

To extend the definition of balanced and almost balanced sequences to non-binary sequences we need the following additional notation. For $\sigma \in \Sigma_q$, let $\#_\sigma(\boldsymbol{x})$ denote the number of occurrences of the symbol σ in $\boldsymbol{x}$. A sequence $\boldsymbol{x} \in \Sigma_q^n$ is called *symbol-balanced* if any symbol $\sigma \in \Sigma_q$ appears in $\boldsymbol{x}$ exactly $\frac{n}{q}$ times. That is, $\#_\sigma(\boldsymbol{x}) = \frac{n}{q}$ for any $\sigma \in \Sigma_q$. When q is even, we say that $\boldsymbol{x}$ is *polarity-balanced* if

$$\sum_{i=0}^{\frac{q}{2}-1} \#_i(\boldsymbol{x}) = \sum_{i=\frac{q}{2}}^{q-1} \#_i(\boldsymbol{x}) = \frac{n}{2}.$$

Definition 6.1 can be extended to *α-almost symbol-balanced* and *α-almost polarity-balanced* as follows.

Definition 6.2. *A sequence $\boldsymbol{x}$ is called* $\varepsilon(n)$-almost symbol-balanced *if for any $\sigma \in \Sigma_q$ we have that*

$$\#_\sigma(\boldsymbol{x}) \in \left[\frac{n}{q} - \varepsilon(n), \frac{n}{q} + \varepsilon(n)\right].$$

Definition 6.3. *A sequence $\boldsymbol{x}$ is called* $\varepsilon(n)$-almost polarity-balanced *if*

$$\left| \sum_{i=0}^{\frac{q}{2}-1} \#_i(\boldsymbol{x}) - \sum_{i=\frac{q}{2}}^{q-1} \#_i(\boldsymbol{x}) \right| \leq 2 \cdot \varepsilon(n).$$

6.2.2 Related Work

In this work we extend our previous work [20] focusing on eliminating windows with small periods. This work utilizes a graph-based reduction technique to establish efficiency and convergence of the construction. Inspired by this, the current study proposes an iterative method for encoding sequences into almost-balanced ones without requiring monotonic progress between the algorithm's steps. In a parallel effort [24, 25], the technique is extended to address diverse constraints. A universal approach is presented and a general methodology for combining constraints is detailed, showcasing the versatility and comprehensive nature of the encoding framework.

6.2.3 Arithmetic Coding

Arithmetic coding [26] serves as a data compression method wherein the encoding process transforms an input sequence into a new sequence, representing a fractional value in the interval $[0, 1)$. Each iteration processes a single symbol from the input, dividing the current interval and designating one of the resulting partitions as the new interval. Consequently, the algorithm progressively operates on smaller intervals, and the output exists within each of these nested intervals.

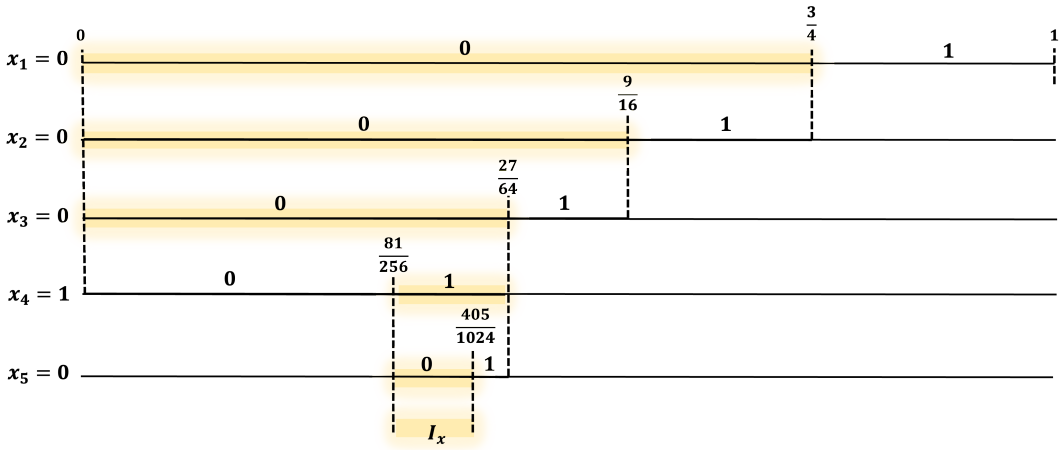


Figure 6.1: Mapping of $\boldsymbol{x} = 00010$ into an interval I_x for $p = \frac{3}{4}$ (and $n = 5$).

One of the main components in our suggested construction is an encoder and decoder pair which are based on binary arithmetic coding. For the completeness of the results in the paper, the key concepts of binary arithmetic coding, which will be used throughout this paper, are described next.[1]

Let $p \in (0,1)$ and let n be an integer. The encoding of a sequence $\boldsymbol{x}$ of length n is done by mapping $\boldsymbol{x}$ into a unique interval $I_{\boldsymbol{x}} \subseteq [0,1)$ as follows.

1) Initialize $I \leftarrow [0,1)$.

2) For $i = 1, 2, \ldots, n$:

   (a) Split the interval I into two sub-intervals, I_L and I_R, of sizes $|I_L| = p \cdot |I|$ and $|I_R| = (1-p) \cdot |I|$.

   (b) If $x_i = 0$, $I \leftarrow I_L$

   (c) Else, $I \leftarrow I_R$

3) $I_{\boldsymbol{x}} = I$.

Finally, the encoding of $\boldsymbol{x}$ is the binary representation[2] of the shortest (fewest number of bits) fraction in the interval $I_{\boldsymbol{x}}$.

After mapping the input to an interval, the output of the algorithm is the binary representation of a fraction within the interval with a minimal number of bits in its binary representation. Given $p \in (0,1)$ and an integer $n > 0$ we denote the corresponding encoder and decoder pair by $f_p^{(\mathbf{ac})} : \Sigma_2^n \to \Sigma_2^*$ and $g_p^{(\mathbf{ac})} : \Sigma_2^* \to \Sigma_2^n$.

[1] This description highlights the details necessary for our derivations and it can be considered a simplification of the standard arithmetic coding technique.

[2] Here we consider the binary representation of a fraction in $[0,1)$ as the binary vector representing the corresponding sum of negative powers of two (similar to the representation of a positive integer, but with negative powers). For example 0.25 is represented by 01 and 0.75 is represented by 11.

Example 6.4. *Figure 6.1 presents the described steps for the mapping of* $\boldsymbol{x} = 00010$ *into an interval* $I_{\boldsymbol{x}}$ *for* $p = \frac{3}{4}$ *(and* $n = 5$*). In this example,*

$$I_{\boldsymbol{x}} = \left[\frac{81}{256}, \frac{405}{1024}\right),$$

and the fraction with the minimal representation within this interval is $0.375 = 2^{-2} + 2^{-3}$*, resulting with the output* 011.

We note that the mapping of a sequence $\boldsymbol{x} \in \Sigma_2^n$ into an interval $I_{\boldsymbol{x}}$ requires splitting the interval n times, where each iteration computes the new interval edges using a single multiplication of two rational numbers. Hence, in the worse-case, both $f_p^{(\mathbf{ac})}$ and $g_p^{(\mathbf{ac})}$ perform $\Theta(n)$ multiplications operations.

6.3 Binary Almost Balanced Sequences

In this section, we discuss the case of binary $\varepsilon(n)$-almost balanced sequences. We first explicitly define the constraint as follows,

$$\mathcal{C}(n, \varepsilon(n)) = \left\{\boldsymbol{x} \in \Sigma_2^n \;\middle|\; w(\boldsymbol{x}) \in \left[\frac{n}{2} - \varepsilon(n), \frac{n}{2} + \varepsilon(n)\right]\right\}.$$

In the next lemma, we show that there exists a single-redundancy-bit construction for $\mathcal{C}(n, \varepsilon(n))$ only if $\varepsilon(n) = \Omega(\sqrt{n})$. More precisely, for $\varepsilon(n) = \alpha\sqrt{n}$, we give lower and upper bounds on the minimal value of α for which there exists a single-redundancy-bit construction for $\mathcal{C}(n, \alpha\sqrt{n})$ for n large enough. More formally, for every $\alpha > 0$ we define $F(n, \alpha) \triangleq \frac{|\mathcal{C}(n, \alpha\sqrt{n})|}{2^n}$. Thus, our goal is to find the minimum α for which there exists n' such that for any $n \geq n'$ we have that $F(n, \alpha) \geq 1/2$.

Lemma 6.5. *There exists a constant* c *such that if* $\alpha \geq c$ *and* n *is large enough, then there exists a single redundancy bit construction for* $\mathcal{C}(n, \alpha\sqrt{n})$*. Otherwise, if* $\alpha < c$ *then there is no such a construction. Moreover, it holds that* $0.335 < c \leq 0.34$*.*

Proof. This result can be seen from the fact that the binomial distribution approaches the normal distribution as $n \to \infty$, with $\mu = n/2$ and $\sigma = \sqrt{n}/2$. Considering the Z-score table [27] we know that at least half of the space is thus contained in

$$[\mu - 0.68\sigma, \mu + 0.68\sigma] = [n/2 - 0.34\sqrt{n}, n/2 + 0.34\sqrt{n}].$$

On the other hand, the interval

$$[\mu - 0.67\sigma, \mu + 0.67\sigma] = [n/2 - 0.335\sqrt{n}, n/2 + 0.335\sqrt{n}]$$

contains strictly less than half of the space. □

Next, we demonstrate an efficient construction with a single redundancy bit for $\alpha > \sqrt{\ln(2)} \approx 0.8325$, inspired by the approach taken in [20] and based on the arithmetic coding [26] technique. To this end, we also define the two following auxiliary constraints,

$$\mathcal{C}_L(n, \alpha\sqrt{n}) = \left\{ \boldsymbol{x} \in \Sigma_2^n \;\middle|\; w(\boldsymbol{x}) \leq \frac{n}{2} + \alpha\sqrt{n} \right\},$$

$$\mathcal{C}_H(n, \alpha\sqrt{n}) = \left\{ \boldsymbol{x} \in \Sigma_2^n \;\middle|\; w(\boldsymbol{x}) \geq \frac{n}{2} - \alpha\sqrt{n} \right\}.$$

Notice that $\mathcal{C}(n, \alpha\sqrt{n}) = \mathcal{C}_L(n, \alpha\sqrt{n}) \cap \mathcal{C}_H(n, \alpha\sqrt{n})$. We now propose the overall construction as follows,

Construction 6.1 (Binary almost-balanced)**.** *Let $\alpha > \sqrt{\ln(2)}$, let n be a sufficiently large*[3] *integer, and let $\boldsymbol{x} \in \{0,1\}^{n-1}$. Our construction is composed of the following two instances of the arithmetic coding described in Section* 6.2.3*:*

- *Binary arithmetic coding with $p_L = 1/2 + \alpha/\sqrt{n} + 1/n$ and a pair of encoder and decoder functions*
$$f_{p_L}^{(\boldsymbol{ac})} : \Sigma_2^n \to \Sigma_2^*,$$
$$g_{p_L}^{(\boldsymbol{ac})} : \Sigma_2^* \to \Sigma_2^n.$$
- *Binary arithmetic coding with $p_H = 1/2 - \alpha/\sqrt{n} - 1/n$ and a pair of encoder and decoder functions*
$$f_{p_H}^{(\boldsymbol{ac})} : \Sigma_2^n \to \Sigma_2^*,$$
$$g_{p_H}^{(\boldsymbol{ac})} : \Sigma_2^* \to \Sigma_2^n.$$

For simplicity, we assume that the output length of $f_{p_L}^{(\boldsymbol{ac})}, f_{p_H}^{(\boldsymbol{ac})}$ is at least $n-2$ (since otherwise, we can pad the output with zeros and we will show that it will be exactly $n-2$). Then, Algorithms 5 *and* 6 *construct an efficient construction with a single redundancy bit and $\mathcal{O}(T(n))$ average time complexity for $T(n)$ the maximum time complexity amongst $f_{p_L}^{(\boldsymbol{ac})}, g_{p_L}^{(\boldsymbol{ac})}, f_{p_H}^{(\boldsymbol{ac})}, g_{p_H}^{(\boldsymbol{ac})}$. That is, the average time complexity is $\mathcal{O}(n\rho)$, where ρ is the time complexity of a single multiplication operation* [4]*.*

The correctness of this construction is stated in the next theorem.

Theorem 6.6. *Construction 6.1 is an efficient construction with a single redundancy bit for $\mathcal{C}(n, \alpha\sqrt{n})$ when $\alpha > \sqrt{\ln(2)}$ and n is large enough.*

The proof of the theorem follows immediately from the following three lemmas.

[3] While the size of n depends on α, the size of n for which the construction work is not too large. For example, for $\alpha = 1$ and $\alpha = 0.835$ we only need $n > 4$ and $n > 6$, respectively

[4] The value ρ depends on the exact number of symbols needed to preserve sufficient accuracy during multiplication.

Algorithm 5: Almost-Balanced Encoder E

Input: $\boldsymbol{x} \in \Sigma_2^{n-1}$.
Output: $\boldsymbol{y} \in \mathcal{C}(n, \alpha\sqrt{n})$.
$\boldsymbol{y} \leftarrow \boldsymbol{x} \circ 0$.
while $\boldsymbol{y} \notin \mathcal{C}(n, \alpha\sqrt{n})$ **do**
  if $\boldsymbol{y} \in \mathcal{C}_L(n, \alpha\sqrt{n})$ **then**
    $\boldsymbol{y} \leftarrow f_{p_L}^{(\mathbf{ac})}(\boldsymbol{y}) \circ 11$.
  end
  else
    $\boldsymbol{y} \leftarrow f_{p_H}^{(\mathbf{ac})}(\boldsymbol{y}) \circ 01$.
  end
end
return $\boldsymbol{y}$.

Algorithm 6: Almost-Balanced Decoder D

Input: $\boldsymbol{y} \in \mathcal{C}(n, \alpha\sqrt{n})$ such that $E(\boldsymbol{x}) = \boldsymbol{y}$ for $\boldsymbol{x} \in \Sigma_2^{n-1}$.
Output: $\boldsymbol{x} \in \Sigma_2^{n-1}$.
while $y_n \neq 0$ **do**
  if $y_{n-1} = 1$ **then**
    $\boldsymbol{y} \leftarrow g_{p_L}^{(\mathbf{ac})}(\boldsymbol{y}_{[1:n-2]})$.
  end
  if $y_{n-1} = 0$ **then**
    $\boldsymbol{y} \leftarrow g_{p_H}^{(\mathbf{ac})}(\boldsymbol{y}_{[1:n-2]})$.
  end
end
return $\boldsymbol{y}_{[1:n-1]}$.

Lemma 6.7. *Algorithm 5 stops with an output* $\boldsymbol{y} \in \mathcal{C}(n, \alpha\sqrt{n})$.

Proof. First, let us prove that in each iteration of Algorithm 5, the length of $\boldsymbol{y}$, is exactly n. Clearly, before entering the while loop for the first time, the length of $\boldsymbol{y}$ is n. Within the while loop, $\boldsymbol{y}$ is modified to be the concatenation of either $f_{p_L}^{(\mathbf{ac})}(\boldsymbol{y})$ or $f_{p_H}^{(\mathbf{ac})}(\boldsymbol{y})$ with two additional bits. Hence, we need to show that the output of $f_{p_L}^{(\mathbf{ac})}(\boldsymbol{y})$ or $f_{p_H}^{(\mathbf{ac})}(\boldsymbol{y})$, respectively, is of length $n-2$. Recall that by our assumption, the output length of $f_{p_L}^{(\mathbf{ac})}, f_{p_H}^{(\mathbf{ac})}$ is always at least $n-2$. Hence, it is sufficient to show that the length cannot be greater than $n-2$.

If $\boldsymbol{y}$ is modified in step 4 then $\boldsymbol{y} \notin \mathcal{C}(n, \alpha\sqrt{n})$ and $\boldsymbol{y} \in \mathcal{C}_L(n, \alpha\sqrt{n})$. That is, $w(\boldsymbol{y}) < \frac{n}{2} - \alpha\sqrt{n}$, and the length of the interval that corresponds to $\boldsymbol{y}$ by the mapping $f_{p_L}^{(\mathbf{ac})}$ is

$$|I_{\boldsymbol{y}}| = \left(\frac{1}{2} - \frac{\alpha}{\sqrt{n}} - \frac{1}{n}\right)^{w(\boldsymbol{y})} \cdot \left(\frac{1}{2} + \frac{\alpha}{\sqrt{n}} + \frac{1}{n}\right)^{n-w(\boldsymbol{y})}.$$

Since this value is minimized when $w(\boldsymbol{y})$ is maximized, we have that any such $\boldsymbol{y}$ is mapped to an interval of length

$$|I_{\boldsymbol{y}}| \geq \left(\frac{1}{2}-\frac{\alpha}{\sqrt{n}}-\frac{1}{n}\right)^{\frac{n}{2}-\alpha\sqrt{n}} \cdot \left(\frac{1}{2}+\frac{\alpha}{\sqrt{n}}+\frac{1}{n}\right)^{\frac{n}{2}+\alpha\sqrt{n}}.$$

Moreover, since

$$\lim_{n\to\infty} 2^{n-2} \cdot \left(\frac{1}{2}-\frac{\alpha}{\sqrt{n}}-\frac{1}{n}\right)^{\frac{n}{2}-\alpha\sqrt{n}} \cdot \left(\frac{1}{2}+\frac{\alpha}{\sqrt{n}}+\frac{1}{n}\right)^{\frac{n}{2}+\alpha\sqrt{n}} = \frac{e^{2\alpha^2}}{4},$$

it can be verified that for any $\alpha > \sqrt{\ln(2)}$ we have that $|I_{\boldsymbol{y}}| \geq 1/2^{n-2}$ for n large enough. This implies that it is possible to enumerate the interval with exactly $n-2$ symbols.

By the definition of Algorithm 5, if the algorithm ends, it stops with a valid output. Hence, it is left to show that the algorithm converges. Similarly to the approach presented in [20], the convergence follows from a reduction to an acyclic graph walk, and it is given here for completeness.

Let $G = (V, \mathcal{E})$ be a directed graph such that $V = \Sigma_2^n$ is the set of nodes and the set of edges $\mathcal{E} \subseteq V \times V$ is defined as follows. From any $\boldsymbol{u} \notin \mathcal{C}(n, \alpha\sqrt{n})$ there is a single outgoing edge to the node $\boldsymbol{v} \in V$, where $\boldsymbol{v} = f_{p_L}^{(\mathbf{ac})}(\boldsymbol{u}) \circ 11$ if $\boldsymbol{y} \in \mathcal{C}_L(n, \alpha\sqrt{n})$, and otherwise $\boldsymbol{v} = f_{p_H}^{(\mathbf{ac})}(\boldsymbol{u}) \circ 01$. That is, there is an edge from node $\boldsymbol{u}$ to node $\boldsymbol{v}$ if $\boldsymbol{v}$ is the unique sequence such that the operations inside the while loop of Algorithm 5 will modify $\boldsymbol{y} = \boldsymbol{u}$ to $\boldsymbol{y} = \boldsymbol{v}$. Note that the mappings $f_{p_L}^{(\mathbf{ac})}$ and $f_{p_H}^{(\mathbf{ac})}$ are invertible functions and hence the in-degree of all the nodes in V is at most one. Moreover, by the definition of $\mathcal{E}$, any node $\boldsymbol{v} \in V$ for which $v_n = 0$, satisfies $d_{in}(\boldsymbol{v}) = 0$, where $d_{in}(\boldsymbol{v})$ is the in-degree of $\boldsymbol{v}$.

Assume by contradiction that the encoder does not converge for an input $\boldsymbol{x} \in \Sigma_2^{n-1}$ and let

$$\boldsymbol{x} \circ 0 = \boldsymbol{y}^{(0)}, \boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots$$

be the list of nodes that correspond to the values of $\boldsymbol{y}$ before each iteration of the while loop in Algorithm 5. Since $|V|$ is finite, the path $\boldsymbol{y}^{(0)} \to \boldsymbol{y}^{(1)} \to \boldsymbol{y}^{(2)} \to \cdots$ contains a cycle, i.e., there is an index i such that

$$\boldsymbol{y}^{(i)} \to \boldsymbol{y}^{(i+1)} \to \cdots \to \boldsymbol{y}^{(j-1)} \to \boldsymbol{y}^{(j)} = \boldsymbol{y}^{(i)}.$$

Since $y_n^{(0)} = 0$, we have that $d_{in}(\boldsymbol{y}^{(0)}) = 0$ and hence $\boldsymbol{y}^{(0)}$ is not on the cycle. Hence, there exists an index i' such that $d_{in}(\boldsymbol{y}^{(i')}) = 2$, which is a contradiction. □

Example 6.8. *Let $n = 8, \alpha = 1$ and $\boldsymbol{x} = 1000000 \in \Sigma_2^7$. Note that*

$$\mathcal{C}(8, \sqrt{8}) = \{\boldsymbol{y} \in \Sigma_2^8 \mid 2 \leq w(\boldsymbol{y}) \leq 6\}.$$

Using the same notations as in the proof of Lemma 6.7, Algorithm 5 begins with

$$\boldsymbol{y}^{(0)} = \boldsymbol{x} \circ 0 = 1000000 \circ 0.$$

As $\boldsymbol{y}^{(0)} \notin \mathcal{C}(8, \sqrt{8})$ *the algorithm enters the while loop and line 4 is executed (*$w(\boldsymbol{y}^{(0)}) \leq 6$*). It can be verified that in this step* $\boldsymbol{y}^{(0)}$ *is mapped to the interval* $[0.97855, 0.99698)$ *which can be represented using* 111111*. Hence* $\boldsymbol{y}^{(1)} = 111111 \circ 11$*. In the second iteration, line* 6 *is executed and* $\boldsymbol{y}^{(2)} = 100000 \circ 01 \in \mathcal{C}(8, \sqrt{8})$*.*

Lemma 6.9. *For any* $\boldsymbol{x} \in \Sigma_2^{n-1}$*, the decoder* D *from Algorithm* 6 *satisfies* $D(E(\boldsymbol{x})) = \boldsymbol{x}$*.*

Lemma 6.10. *The average number of iterations in the while loop of* E, D *is at most* $|\Sigma| = O(1)$*.*

The proofs of Lemma 6.9 and Lemma 6.10 follow from the reduction of the encoder to a graph walk presented in the proof of Lemma 6.7. Since the proofs are very similar to the ones presented in [24], they are omitted.

6.4 Extensions to Non-Binary Alphabets

In this section we discuss the extension of Construction 6.1 to the non-binary case.

6.4.1 Almost Polarity-Balanced

Construction 6.1 can be modified to obtain almost polarity-balanced sequences for any alphabet of even size. First, we formally define the constraint

$$\mathcal{C}_q^{(\mathbf{pb})}(n, \varepsilon(n)) = \left\{ \boldsymbol{x} \in \Sigma_q^n \;\middle|\; \sum_{i=0}^{\frac{q}{2}-1} \#_i(\boldsymbol{x}) \in \left[\frac{n}{2} - \varepsilon(n), \frac{n}{2} + \varepsilon(n) \right] \right\}.$$

Similarly to the binary case, we are interested in a single redundancy symbol codes and in the next lemma we show that $\varepsilon(n) = \Omega(\sqrt{n})$. To this end, we define $F^{(\mathbf{pb})}(n, \alpha) \triangleq \frac{|\mathcal{C}_q^{(\mathbf{pb})}(n, \alpha\sqrt{n})|}{q^n}$.

Lemma 6.11. *Let* $c = \liminf\{\alpha | F^{(\boldsymbol{pb})}(n, \alpha) \geq 1/q\}$*. Then, it holds that* $c \leq 0.34$*.*

Proof. It holds that

$$\begin{aligned} F^{(\mathbf{pb})}(n, \alpha) &= \frac{|\mathcal{C}_q^{(\mathbf{pb})}(n, \alpha\sqrt{n})|}{q^n} \\ &= \frac{|\mathcal{C}(n, \alpha\sqrt{n})| \cdot (q/2)^n}{q^n} \\ &= \frac{|\mathcal{C}(n, \alpha\sqrt{n})|}{2^n} = F(n, \alpha). \end{aligned}$$

The latter together with Lemma 6.5 implies the claim of the lemma. □

q	Lower bound	Upper bound
2	0.335	0.34
3	0.215	0.22
4	0.155	0.16
5	0.125	0.13
6	0.105	0.11
7	0.09	0.095

Table 6.1: Bounds on $c = \liminf\{\alpha | F^{(\mathbf{pb})}(n,\alpha) \geq 1/q\}$.

While the bound in the previous lemma is not tight, knowing that $F^{(\mathbf{pb})}(n,\alpha) = F(n,\alpha)$, we can derive tighter upper and lower bounds using the same techniques as in the proof of Lemma 6.5. Table 6.1 summarizes the upper and lower bounds on c for small alphabet size, that can be derived by this manner.

Similarly to Construction 6.1, we define $\mathcal{C}_q^{(\mathbf{pb})}(n,\alpha\sqrt{n})$ as the intersection of the following two constraint channels,

$$\mathcal{C}_{q,L}^{(\mathbf{pb})}(n,\alpha\sqrt{n}) = \left\{ \boldsymbol{x} \in \Sigma_q^n \;\middle|\; \sum_{i=0}^{\frac{q}{2}-1} \#_i(\boldsymbol{x}) \leq \frac{n}{2} + \alpha\sqrt{n} \right\},$$

$$\mathcal{C}_{q,H}^{(\mathbf{pb})}(n,\alpha\sqrt{n}) = \left\{ \boldsymbol{x} \in \Sigma_q^n \;\middle|\; \sum_{i=0}^{\frac{q}{2}-1} \#_i(\boldsymbol{x}) \geq \frac{n}{2} - \alpha\sqrt{n} \right\}.$$

To address non-binary alphabets, we modify our arithmetic coding based mappings as follows. First, in each iteration we partition the current interval into q sub-intervals (instead of two) and if the current symbol is i, we continue with the i-th sub-interval to the next iteration. For $p \in (0,1)$ we define the size of the first $\frac{q}{2}$ sub-intervals to be $\frac{2p}{q}|I|$, and the size of the last $\frac{q}{2}$ sub-intervals to be $\frac{2(1-p)}{q}|I|$, where I is the current interval. Given $p \in (0,1)$, an integer $n > 0$, and an even integer $q \geq 2$, we denote the corresponding encoder and decoder pair by $f_{q,p}^{(\mathbf{pb\text{-}ac})} : \Sigma_q^n \to \Sigma_q^*$ and $g_{q,p}^{(\mathbf{pb\text{-}ac})} : \Sigma_q^* \to \Sigma_q^n$.

Construction 6.2 (Almost polarity balanced)**.** *For an even integer $q \geq 2$, let $\alpha > \sqrt{\ln(q)}$, let n be a sufficiently large integer, and let $\boldsymbol{x} \in \Sigma_q^{n-1}$. Our construction is composed of the following two instances of the modified arithmetic coding:*

- *q-ary arithmetic coding with $p_L = 1/2 + \alpha/\sqrt{n} + 1/n$ and a pair of encoder and decoder functions*
$$f_{q,p_L}^{(\mathbf{pb\text{-}ac})} : \Sigma_q^n \to \Sigma_q^*,$$
$$g_{q,p_L}^{(\mathbf{pb\text{-}ac})} : \Sigma_q^* \to \Sigma_q^n.$$
- *q-ary arithmetic coding with $p_H = 1/2 - \alpha/\sqrt{n} - 1/n$ and a pair of encoder and decoder functions*
$$f_{q,p_H}^{(\mathbf{pb\text{-}ac})} : \Sigma_q^n \to \Sigma_q^*,$$

$$g_{q,p_H}^{(\textbf{pb-ac})} : \Sigma_q^* \to \Sigma_q^n.$$

Then Algorithms 5 *and* 6 *(with the corresponding modifications) construct an efficient construction with a single redundancy symbol and* $\mathcal{O}(T(n))$ *average time complexity, where* $T(n)$ *is the maximum complexity amongst* $f_{q,p_L}^{(\textbf{pb-ac})}, g_{q,p_L}^{(\textbf{pb-ac})}, f_{q,p_H}^{(\textbf{pb-ac})}, g_{q,p_H}^{(\textbf{pb-ac})}$.

The correctness of Construction 6.2 follows from the same arguments that were presented in the proof of Construction 6.1. Hence, it is sufficient to show that the length of $\boldsymbol{y}$ before each iteration of the while-loop is n. That is, we need to show that

$$\left(\frac{1}{q}-\frac{2\alpha}{q\sqrt{n}}-\frac{2}{qn}\right)^{\frac{n}{2}-\alpha\sqrt{n}}\cdot\left(\frac{1}{q}+\frac{2\alpha}{q\sqrt{n}}+\frac{2}{qn}\right)^{\frac{n}{2}+\alpha\sqrt{n}} \geq \frac{1}{q^{n-2}},$$

which holds for any $\alpha > \sqrt{\ln(q)}$. We note that for $q = 2$ Construction 6.2 is identical to Construction 6.1. However, for $q > 2$ we can improve Construction 6.2 by noticing that now instead of using two bits it is enough to use one symbol in order to indicate the three options of determining when the decoder stops and whether to decode with $g_{q,p_L}^{(\textbf{pb-ac})}$ or $g_{q,p_H}^{(\textbf{pb-ac})}$. Hence, we can change Algorithm 5 such that in step 4 we assign $f_{q,p_L}^{(\textbf{pb-ac})}(\boldsymbol{y}) \circ 1$ into $\boldsymbol{y}$ and in step 7 we assign $f_{q,p_L}^{(\textbf{pb-ac})}(\boldsymbol{y}) \circ 2$ into $\boldsymbol{y}$. Accordingly, we modify step 2 and step 5 of Algorithm 6 to check whether y_n equals 1 or 2, respectively. By doing so, we allow the output of $f_{q,p_L}^{(\textbf{pb-ac})}$ and $f_{q,p_H}^{(\textbf{pb-ac})}$ to be of length $n-1$. Hence, we need

$$\left(\frac{1}{q}-\frac{2\alpha}{q\sqrt{n}}-\frac{2}{qn}\right)^{\frac{n}{2}-\alpha\sqrt{n}}\cdot\left(\frac{1}{q}+\frac{2\alpha}{q\sqrt{n}}+\frac{2}{qn}\right)^{\frac{n}{2}+\alpha\sqrt{n}} \geq \frac{1}{q^{n-1}},$$

which holds for $\alpha > \sqrt{\frac{\ln(q)}{2}}$.

A more thorough examination, reveals that the latter construction can be further improved for larger values of q. For such values of q, we do not need $f_{q,p_L}^{(\textbf{pb-ac})}$ and $f_{q,p_H}^{(\textbf{pb-ac})}$ to compress the input at all. For our purposes, it is sufficient to only restrict the values of the last symbol in the output and use the remaining symbols to distinguish between the states.

6.4.2 Almost Symbol-Balanced

Lastly, we discuss $\varepsilon(n)$-almost symbol-balanced q-ary sequences. For simplicity, we only give a construction for $q = 4$ while the generalization to $q = 2^\ell$ for any positive integer ℓ is straightforward. Given n and ε we define the constraint as,

$$\mathcal{C}_4^{(\textbf{sb})}(n,\varepsilon(n)) = \left\{\boldsymbol{x} \in \Sigma_4^n \;\middle|\; \#_\sigma(\boldsymbol{x}) \in \left[\frac{n}{4}-\varepsilon(n), \frac{n}{4}+\varepsilon(n)\right], \forall \sigma \in \Sigma_4\right\}.$$

We start by noting that any $\varepsilon(n)$-almost symbol-balanced sequence is also a $\varepsilon(n)$-almost polarity-balanced. Hence, our analysis of almost polarity-balanced codes implies that if a single-redundancy-symbol $\varepsilon(n)$-almost symbol-balanced code exists then $\varepsilon(n) = \Omega(\sqrt{n})$. Therefore, we focus again on the case where $\varepsilon(n) = \alpha\sqrt{n}$.

Our construction is based on defining a subset of $\mathcal{C}_4^{(\mathbf{sb})}(n, \alpha\sqrt{n}))$ as the intersection of the following three $\frac{\alpha\sqrt{n}}{2}$-polarity-balanced 4-ary codes,

$$\mathcal{C}_{0,i}^{(\mathbf{pb})} = \left\{ \boldsymbol{x} \in \Sigma_4^n \;\middle|\; \#_0(\boldsymbol{x}) + \#_i(\boldsymbol{x}) \in \left[\frac{n}{2} - \frac{\alpha\sqrt{n}}{2}, \frac{n}{2} + \frac{\alpha\sqrt{n}}{2} \right] \right\},$$

for $i \in \{1, 2, 3\}$.

Lemma 6.12. *It holds that*

$$\mathcal{C}_{0,1}^{(\boldsymbol{pb})} \cap \mathcal{C}_{0,2}^{(\boldsymbol{pb})} \cap \mathcal{C}_{0,3}^{(\boldsymbol{pb})} \subseteq \mathcal{C}_4^{(\boldsymbol{sb})}(n, \alpha\sqrt{n}).$$

Construction 6.3 (Almost symbol balanced)**.** *Let $\alpha > 2\sqrt{\ln(4)}$, let n be a sufficiently large integer, and let $\boldsymbol{x} \in \Sigma_q^{n-1}$. Our construction is composed of three pairs of the modified arithmetic coding that were utilized in Construction 6.2. For each $i \in \{1, 2, 3\}$ consider $\mathcal{C}_{0,i}^{(\boldsymbol{pb})}$ and define:*

- *4-ary arithmetic coding with $p_L = 1/2 + \alpha/2\sqrt{n} + 1/n$ that associates the first two intervals in each partition with 0 and i, and a pair of encoder and decoder functions*

$$f_{4,p_L,i}^{(\boldsymbol{pb\text{-}ac})} : \Sigma_4^n \to \Sigma_4^*$$
$$g_{4,p_L,i}^{(\boldsymbol{pb\text{-}ac})} : \Sigma_4^* \to \Sigma_4^n.$$

- *4-ary arithmetic coding with $p_H = 1/2 - \alpha/2\sqrt{n} - 1/n$ that associates the first two intervals in each partition with 0 and i, and a pair of encoder and decoder functions*

$$f_{4,p_H,i}^{(\boldsymbol{pb\text{-}ac})} : \Sigma_4^n \to \Sigma_4^*$$
$$g_{4,p_H,i}^{(\boldsymbol{pb\text{-}ac})} : \Sigma_4^* \to \Sigma_4^n.$$

- *The constraint channels*

$$\mathcal{C}_{L,0,i}^{(\boldsymbol{pb})} = \left\{ \boldsymbol{x} \in \Sigma_4^n \;\middle|\; \#_0(\boldsymbol{x}) + \#_i(\boldsymbol{x}) \leq \frac{n}{2} + \frac{\alpha\sqrt{n}}{2} \right\}$$
$$\mathcal{C}_{H,0,i}^{(\boldsymbol{pb})} = \left\{ \boldsymbol{x} \in \Sigma_4^n \;\middle|\; \#_0(\boldsymbol{x}) + \#_i(\boldsymbol{x}) \geq \frac{n}{2} - \frac{\alpha\sqrt{n}}{2} \right\}$$

Then Algorithms 7 and 8 construct an efficient construction with a single redundancy symbol and $\mathcal{O}(T(n))$ average time complexity, where $T(n)$ is the maximum complexity amongst $f_{4,p_L,i}^{(\boldsymbol{pb\text{-}ac})}, f_{4,p_H,i}^{(\boldsymbol{pb\text{-}ac})}, g_{4,p_L,i}^{(\boldsymbol{pb\text{-}ac})}, g_{4,p_H,i}^{(\boldsymbol{pb\text{-}ac})}$ for $i \in \{1, 2, 3\}$.

The correctness of Construction 6.3 follows from arguments similar to the those presented in the proof of Construction 6.1, and the observation that for $i \in \{1, 2, 3\}$ the output of $f_{4,p_H,i}^{(\mathbf{pb\text{-}ac})}$ and $f_{4,p_H,i}^{(\mathbf{pb\text{-}ac})}$ is of length $n - 2$ for $\alpha > 2\sqrt{\ln(4)}$.

Algorithm 7: Almost-Balanced 4-ary Encoder E_4

Input: $x \in \Sigma_4^{n-1}$.
Output: $y \in \mathcal{C}_4^{(\mathbf{sb})}(n, \alpha\sqrt{n})$.
$y \leftarrow x \circ 0$.
while $y \notin \mathcal{C}_4^{(sb)}(n, \alpha\sqrt{n})$ **do**
  for $i \in \{1, 2, 3\}$ **do**
    if $y \notin \mathcal{C}_{0,i}^{(pb)}$ **then**
      if $y \notin \mathcal{C}_{L,0,i}^{(pb)}$ **then**
        $y \leftarrow f_{4,p_L,i}^{(\mathbf{pb\text{-}ac})}(y) \circ 1 \circ i$.
      end
      else
        $y \leftarrow f_{4,p_H,i}^{(\mathbf{pb\text{-}ac})}(y) \circ 0 \circ i$.
      end
    end
    break
  end
end
return y.

6.5 Conclusion

This work studies the problem of encoding almost-balanced sequences using a single redundancy symbol. Our constructions achieve an optimally balanced order of $\Theta(\sqrt{n})$, though there remains a multiplicative gap between the theoretical bounds on $\varepsilon(n)$ and the values applicable in our constructions. Future work should focus on analyzing the number of symbols needed to maintain sufficient precision in the multiplication of rational numbers. This would enable a more accurate evaluation of our method's average time and memory complexities and facilitate a detailed comparison with the enumerative coding approach. Additionally, bounding the worst-case time complexity of the algorithms remains a challenging problem for future investigation. Experimental results verified that the number of iterations of Algorithm 5 is at most 7 for words of length $n = 20$.

Algorithm 8: Almost-Balanced 4-ary Decoder D_4

Input: $\boldsymbol{y} \in \mathcal{C}_4^{(\mathbf{sb})}(n, \alpha\sqrt{n})$ such that $E_4(\boldsymbol{x}) = \boldsymbol{y}$ for some $\boldsymbol{x} \in \Sigma_4^{n-1}$.
Output: $\boldsymbol{x} \in \Sigma_4^{n-1}$.
while $y_n \neq 0$ **do**
  if $y_{n-1} = 1$ **then**
    $\boldsymbol{y} \leftarrow g_{4,p_L,y_n}^{(\mathbf{pb\text{-}ac})}(\boldsymbol{y}_{[1:n-2]})$.
  end
  if $y_{n-1} = 0$ **then**
    $\boldsymbol{y} \leftarrow g_{4,p_H,y_n}^{(\mathbf{pb\text{-}ac})}(\boldsymbol{y}_{[1:n-2]})$.
  end
end
return $\boldsymbol{y}_{[1:n-1]}$.

Bibliography

[1] D. Knuth, "Efficient balanced codes," *IEEE Transactions on Information Theory*, vol. 32, no. 1, pp. 51–53, 1986.

[2] M. Blawat, K. Gaedke, I. Huetter, X.-M. Chen, B. Turczyk, S. Inverso, B. W. Pruitt, and G. M. Church, "Forward error correction for DNA data storage," *Procedia Computer Science*, vol. 80, pp. 1011–1022, 2016.

[3] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angewandte Chemie Int. Edition*, no. 8, pp. 2552–2555, 2015.

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

[5] S. M. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Scientific reports*, vol. 7, no. 1, p. 5-11, 2017.

[6] S. M. H. T. Yazdi, H. M. Kiah, E. Garcia-ruiz, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: trends and methods," *IEEE Transactions on Molecular, Biological and Multi-Scale Communications* vol. 1, no. 3, pp. 230–248, 2015.

[7] L. G. Tallini and B. Bose, "Balanced codes with parallel encoding and decoding," *IEEE Transactions on Computers*, vol. 48, no. 8, pp. 794–814, 1999.

[8] L. G. Tallini, R. M. Capocelli, and B. Bose, "Design of some new efficient balanced codes," *IEEE Transactions on Information Theory*, vol. 42, no. 3, pp. 790–802, 1996.

[9] J. H. Weber and K. A. S. Immink, "Knuth's balanced codes revisited," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1673–1679, 2010.

[10] K. A. S, Immink and J. H. Weber, "Very efficient balanced codes," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 188–192, 2010.

[11] L. G. Tallini, and U. Vaccaro, "Efficient m-ary balanced codes," *Discrete Applied Mathematics*, vol. 92, no. 1, pp. 17–56, 1999.

[12] R. Mascella and L. G. Tallini, "On symbol permutation invariant balanced codes," in *2005 IEEE International Symposium on Information Theory (ISIT)*, 2005, pp. 2100–2104.

[13] R. Mascella and L. G. Tallini, "Efficient m-ary balanced codes which are invariant under symbol permutation," *IEEE Transactions on Computers*, vol. 55, no. 8, pp. 929–946, 2006.

[14] T. G. Swart and J. H. Weber, "Efficient balancing of q-ary sequences with parallel decoding," *2009 IEEE International Symposium on Information Theory (ISIT)*, 2009, pp. 1564–1568.

[15] J. H. Weber, K. A. S. Immink, P. H. Siegel, and T. G. Swart, "Polarity-balanced codes," in *2013 Information Theory and Applications Workshop (ITA)*, 2013, pp. 1–5.

[16] K. A. S. Immink, J. H. Weber, and H. C. Ferreira, "Balanced runlength limited codes using Knuth's algorithm," in *2011 IEEE International Symposium on Information Theory (ISIT)*, 2011, pp. 317–320.

[17] T. M. Cover, "Enumerative Source Encoding," *IEEE Transactions on Information Theory*, vol. 19, no. 1, pp. 73–77, 1973.

[18] B. Ryabko, "A general method for the development of constrained codes," *arXiv preprint arXiv:2405.14570*, 2024.

[19] T. T. Nguyen, K. Cai, and K. A. S. Immink, "Binary subblock energy-constrained codes: Knuth's balancing and sequence replacement techniques," in *2020 IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 37–41.

[20] A. Kobovich, O. Leitersdorf, D. Bar-Lev, and E. Yaakobi, "Codes for constrained periodicity," in *2022 IEEE International Symposium on Information Theory and its Applications (ISITA)*, 2022.

[21] A. J. Van Wijngaarden and K. A. S. Immink, "On the construction of constrained codes employing sequence replacement techniques," in *1997 IEEE International Symposium on Information Theory (ISIT)*, 1997, p. 144.

[22] C. Schoeny, A. Wachter-Zeh, R. Gabrys, and E. Yaakobi, "Codes correcting a burst of deletions or insertions," *IEEE Transactions on Information Theory*, vol. 63, no. 4, pp. 1971–1985, 2017.

[23] A. J. Van Wijngaarden and K. A. S. Immink, "Construction of maximum run-length limited codes using sequence replacement techniques," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 200–207, 2010.

[24] A. Kobovich, O. Leitersdorf, D. Bar-Lev, and E. Yaakobi, "Universal framework for parametric constrained coding," *2024 IEEE International Symposium on Information Theory (ISIT)*, 2024.

[25] D. Bar-Lev, A. Kobovich, O. Leitersdorf, and E. Yaakobi, "Universal framework for parametric constrained coding," *arXiv preprint arXiv:2304.01317*, 2023.

[26] J. Rissanen and G. G. Langdon, "Arithmetic coding," *IBM Journal of Research and Development*, vol. 23, no. 2, pp. 149–162, 1979.

[27] E. Kreyszig, H. Kreyszig, and E. Norminton *Advanced engineering mathematics,* 10th ed., John Wiley & Sons, 2011.

Part III

Towards Practical DNA Storage Systems

Chapter 7

Representing Information on DNA using Patterns Induced by Enzymatic Labeling

Daniella Bar-Lev, Tuvi Etzion, Eitan Yaakobi, and Zohar Yakhini

Abstract

Enzymatic DNA labeling is a powerful tool with applications in biochemistry, molecular biology, biotechnology, medical science, and genomic research. This paper contributes to the evolving field of DNA-based data storage by presenting a formal framework for modeling DNA labeling in strings, specifically tailored for data storage purposes. Our approach involves a known DNA molecule as a template for labeling, employing patterns induced by a set of designed labels to represent information. One hypothetical implementation can use CRISPR-Cas9 and gRNA reagents for labeling. Various aspects of the general labeling channel, including fixed-length labels, are explored, and upper bounds on the maximal size of the corresponding codes are given. The study includes the development of an efficient encoder-decoder pair that is proven optimal in terms of maximum code size under specific conditions.

7.1 Introduction

Enzymatic DNA labeling is a powerful tool with applications in biochemistry, molecular biology, biotechnology, medical science, and genomic research [1–6]. The technique involves the deliberate labeling of specific parts of the DNA molecule with specific markers. In recent literature, it was also demonstrated that DNA labeling (and more generally, DNA editing) can be used in the emerging technology of data storage on DNA. see e.g., [7, 8]. One important approach to labeling is the use of CRISPR-Cas9 based systems [2, 9–11].

Beyond its practical applications, recent studies shifted focus towards understanding DNA labeling from an information theoretic perspective. In [12] Hanania et. al. modeled the process

of DNA labeling as a communication channel. In this setup, either a fixed single label or a small set of fixed labels is being used. The input is a 4-ary sequence representing the DNA molecule and the output is a sequence in which *non-zero* entries represent the presence of labels in the molecule. The work presents multiple results regarding the capacity of this channel, as a function of label(s) used. Another related communication channel, which was studied by Nogin et. al. [13], considered the process of Optical Genome Mapping (OGM) [14, 15], a useful application of DNA labeling that involves optically imaging DNA fragments containing labeled short sequence patterns. The study utilizes techniques from information theory to enable the design of optimal labeling patterns for this application. In [7] the authors introduced and investigated concept related to native DNA to store information using detectable chemical modifications.

In this work, we define a formal framework for modeling labeling in DNA strings and present results pertaining to optimal design in such systems, for the purpose of data storage in DNA. In contrast to the model suggested in [12], in our model a specific known DNA molecule is considered as a template for labeling, while the information is represented by patterns induced by a set of labels designed for this purpose. More specifically, we code any given message to a set of labels that will induce a unique pattern. When reading the information, post communication, the output of the channel is a binary vector representing the sites labeled by the selected labels in the known template DNA molecule.

We characterize the performance of this labeling channel addressing several possible variants. In particular, we consider labels of fixed length ℓ, which is the expected case if CRISPR-Cas9 were to be used, as well as systems that can support a dynamic range of lengths. We developed, for example, an efficient encoder and decoder pair for this channel in the special case in which ℓ is fixed and the template sequence is ℓ-*repeat-free*. In this case, we proved optimality of our construction.

The rest of the paper is organized as follows. In Section 7.2, we introduce the definitions that will be utilized throughout the paper, including the formal definitions of our labeling model and the problems investigated. Additionally, Section 7.2 provides several bounds that consider strings with a small period as templates. Moving on to Section 7.3, our focus shifts to labels with a fixed length, denoted as ℓ. This section presents both upper and lower bounds on the maximum size of codes in our model. Furthermore, we introduce an efficient construction that achieves the upper bound. Finally, Section 7.4 concludes the paper, briefly summarizing some of our key findings and suggesting potential future directions.

7.2 Definitions, Problem Statement, and a First Bound

7.2.1 Definitions

Let $\Sigma = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$ denote the DNA alphabet, and let Σ^n denote the set of all length-n sequences over Σ and Σ^* is the set of all sequences of any length over Σ. Denote by $[n]$ the set $\{1, 2, \ldots, n\}$. For a sequence $S = (s_1, \ldots, s_n) \in \Sigma^n$, and $1 \leq i \leq n - \ell + 1$, let $S_{[i;\ell]} \triangleq (s_i, \ldots, s_{i+\ell-1})$ and let $W_\ell(S) \triangleq \{S_{[i;\ell]} : 1 \leq i \leq n - \ell + 1\}$. A *label* $\lambda \in \Sigma^*$ is a relatively short sequence over Σ. For a sequence $S \in \Sigma^n$, the *labeling* of S with the label λ is defined as follows.

Definition 7.1. *Given a label $\lambda \in \Sigma^{\ell}$, the* λ-labeling function, *$\lambda : \Sigma^* \to \{0,1\}^*$, is a mapping that corresponds to the label λ, where the* λ-labeling *of a sequence $S \in \Sigma^n$ is defined as a binary sequence $\lambda(S) \in \{0,1\}^n$ in which $\lambda(S)_{[i:\ell]} = (1,1,\ldots,1)$ if and only if $S_{[i:\ell]} = \lambda$.*

Example 7.2. *Let $\lambda =$ AAAC $\in \Sigma^4$. The λ-labeling of the two sequences $S_1, S_2 \in \Sigma^{20}$ is presented below.*

$$\begin{aligned} S_1 &= \mathsf{A\,A\,A\,A\,A\,C\,T\,G\,T\,G\,C\,A\,T\,A\,A\,A\,A\,C\,C\,G} \\ \lambda(S_1) &= 0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,0 \end{aligned}$$

$$\begin{aligned} S_2 &= \mathsf{A\,A\,A\,C\,A\,A\,A\,C\,T\,C\,C\,G\,G\,A\,T\,G\,T\,G\,G\,A} \\ \lambda(S_2) &= 1\,1\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \end{aligned}$$

In this work, we will be interested in labeling using multiple labels. Let $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_t\}$ be a set of t labels $\lambda_i \in \Sigma^*$. The Λ-labeling function corresponds to labeling with all the t labels in Λ together and is defined as follows.

Definition 7.3. *Let $\Lambda = \{\lambda_1, \ldots, \lambda_t\}$ be a set of t labels $\lambda_i \in \Sigma^{\ell_i}$. The* Λ-labeling function, *$\Lambda : \Sigma^* \to \{0,1\}^*$, is a mapping function that corresponds to the λ-labeling with all the labels $\lambda \in \Lambda$. The Λ-labeling of a sequence $S \in \Sigma^n$ is defined as the bitwise OR of the t binary sequences $\lambda_i(S)$, $i \in [t]$, i.e.,*

$$\Lambda(S) \triangleq \lambda_1(S) \vee \lambda_2(S) \vee \cdots \vee \lambda_t(S),$$

where $\vee$ stands for the bitwise OR operation.

In other words, for a set of t labels $\Lambda = \{\lambda_1, \ldots, \lambda_t\}$, the Λ-labeling of a sequence $S \in \Sigma^n$ is equal to the binary sequence $\Lambda(S) \in \{0,1\}^n$ in which $\Lambda(S)_{[i:\ell_j]} = (1\,1\,\cdots\,1)$ if and only if $S_{[i:\ell_j]} = \lambda_j \in \Lambda$ for some $j \in [t]$.

Example 7.4. *Let $\Lambda = \{$AAAC, CC, GTG$\}$. The Λ-labeling of the two sequences $S_1, S_2 \in \Sigma^{20}$ is given as follow.*

$$\begin{aligned} S_1 &= \mathsf{A\,A\,A\,A\,A\,C\,T\,G\,T\,G\,C\,A\,T\,A\,A\,A\,A\,C\,C\,G} \\ \Lambda(S_1) &= 0\,0\,1\,1\,1\,1\,0\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1\,1\,0 \end{aligned}$$

$$\begin{aligned} S_2 &= \mathsf{A\,A\,A\,C\,A\,A\,A\,C\,T\,C\,C\,G\,G\,A\,T\,G\,T\,G\,G\,A} \\ \Lambda(S_2) &= 1\,1\,1\,1\,1\,1\,1\,1\,0\,1\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0 \end{aligned}$$

Definition 7.5. *A* labeling code *$\mathcal{C}$ is a collection of sets of labels $\{\Lambda_i\}_{i=1}^{M}$ of sizes $\{t_i\}_{i=1}^{M}$ wherein each $\Lambda_i \in \mathcal{C}$ is called a* codeset. *Given a sequence $S \in \Sigma^*$, we say that a labeling code $\mathcal{C}$ is* S-uniquely-decodable *if for any two distinct codesets $\Lambda_1, \Lambda_2 \in \mathcal{C}$, we have that $\Lambda_1(S) \neq \Lambda_2(S)$.*

We note that Definition 7.5 is equivalent to saying that a labeling code $\mathcal{C}$ is S-uniquely-decodable if any codeset $\Lambda \in \mathcal{C}$ can be uniquely recovered given S and the Λ-labeling of S, $\Lambda(S)$.

Definition 7.6. *Let Λ_1, Λ_2 be two sets and let $S \in \Sigma^*$ be a sequence. We say that Λ_1, Λ_2 are* S-equivalent *and denote $\Lambda_1 \equiv_S \Lambda_2$ if the labeling of S with Λ_1 is identical to the labeling of S with Λ_2, i.e., $\Lambda_1(S) = \Lambda_2(S)$.*

Example 7.7. *Consider the code $\mathcal{C} = \{\Lambda_1, \Lambda_2, \Lambda_3\}$, where*

$$\Lambda_1 = \{\mathsf{AC}\}, \quad \Lambda_2 = \{\mathsf{AC}, \mathsf{GT}\}, \quad \Lambda_3 = \{\mathsf{A}, \mathsf{C}\}.$$

It can be verified that $\mathcal{C}$ is S-uniquely-decodable for the sequence $S = \mathsf{ACGTAAAT}$, by observing that

$$\begin{aligned} \Lambda_1(S) &= 11000000, \\ \Lambda_2(S) &= 11110000, \\ \Lambda_3(S) &= 11001110, \end{aligned}$$

are all different. On the other hand, $\mathcal{C}$ is not S'-uniquely-decodable for the sequence $S' = \mathsf{ACGTACGT}$, since $\Lambda_1(S') = 11001100 = \Lambda_3(S')$, i.e., Λ_1 and Λ_3 are S'-equivalent.

7.2.2 Problems Statement

In order to represent information using DNA labeling, we need a large codebook $\mathcal{C}$ which is S-uniquely-decodable, for some *reference sequence* $S \in \Sigma^n$. To this end, we need to consider both the design of such codes as well as the selection of the reference sequence S. These objectives are formally stated as follows.

Problem 7.1. *Given a reference sequence $S \in \Sigma^n$, find the maximum size of a labeling code $\mathcal{C}$ which is S-uniquely-decodable and efficient encoder and decoder algorithms for an S-uniquely-decodable code $\mathcal{C}_S$ that achieves this maximum value. That is, we want to find the value*

$$M(S) \triangleq \max\{\, |\mathcal{C}| \;:\; \mathcal{C} \text{ is } S\text{-uniquely-decodable}\},$$

and efficient encoder and decoder for an S-uniquely-decodable code $\mathcal{C}_S$ such that

$$|\mathcal{C}_S| = M(S).$$

Problem 7.2. *Given a collection of T labels $\mathcal{V} = \{\lambda_1, \lambda_2, \ldots, \lambda_T\}$, referred as the* executable labels*, find the following.*

1) Given a reference sequence $S \in \Sigma^n$, find the value

$$M(S, \mathcal{V}) \triangleq \max\left\{ |\mathcal{C}| \;\middle|\; \begin{array}{c} \mathcal{C} \text{ is } S\text{-uniquely-decodable,} \\ \mathcal{C} \subseteq \mathcal{P}(\mathcal{V}) \end{array} \right\},$$

where $\mathcal{P}(\mathcal{V})$ is the power set of $\mathcal{V}$ (i.e., for any $\Lambda \in \mathcal{C}$ we have $\Lambda \subseteq \mathcal{V}$).

2) Given a positive integer n find the value

$$M(n, \mathcal{V}) \triangleq \max_{S \in \Sigma^n} \{M(S, \mathcal{V})\},$$

a reference sequence $S_{(n,\mathcal{V})} \in \Sigma^n$ such that

$$M(S_{(n,\mathcal{V})}, \mathcal{V}) = M(n, \mathcal{V}),$$

and an $S_{(n,\mathcal{V})}$-uniquely decodable code $\mathcal{C}_{(n,\mathcal{V})} \subseteq \mathcal{P}(\mathcal{V})$ with efficient encoder and decoder algorithms.

Problem 7.1 represents a theoretical question and we will describe some related results. Problem 7.2 represents a variation of the model that encompasses the case of CRISPR labels [2, 9–11] where λ_i will be the corresponding potential edit sites.

7.2.3 Basic Results using Periodicity

We start with the simplest case of Problem 7.1 in which the reference sequence S consists of a single run. That is, $S = \sigma^n$ for some $\sigma \in \Sigma$ and positive integer n. For this case, we fully solve Problem 7.1 in the next lemma.

Lemma 7.8. *If S is a sequence with a single run of the symbol σ, then for any S-uniquely-decodable labeling code $\mathcal{C}$ we have that $M(S) = 2$. Furthermore, the code $\mathcal{C}_\sigma = \{\varnothing, \{\sigma\}\}$, is S-uniquely-decodable.*

Proof. To see that $\mathcal{C}_\sigma$ is S-uniquely-decodable note that for $\Lambda_1 = \varnothing$ and $\Lambda_2 = \{\sigma\}$ we have that $\Lambda_1(S)$ is the *all-zero* word while $\Lambda_2(S)$ is the *all-one* word.

Let $\mathcal{C}$ be an S-uniquely-decodable labeling code with maximum size. Note that for any label $\lambda \in \Sigma^*$, if $\lambda \neq \sigma^i$, for any integer $0 < i \leq |S|$, then λ is not a substring of S and $\lambda(S)$ is the *all-zero* word. Otherwise, we have that $\lambda(S)$ is the *all-one* word. By definition of the bitwise OR operation, for any codeset $\Lambda \in \mathcal{C}$ we have that either $\Lambda \equiv_S \varnothing$ or $\Lambda \equiv_S \{\sigma\}$. Since $\mathcal{C}$ is S-uniquely-decodable, the latter implies that $|\mathcal{C}| \leq 2$. □

To extend Lemma 7.8 to more involved cases, we first present the definition of a *period*.

Definition 7.9. *For $S \in \Sigma^n$, $1 \leq \pi \leq n-1$ is called a* period *of S if $\pi | n$ and for all $1 \leq i \leq n - \pi + 1$, $S_i = S_{i+\pi}$. Additionally, we let $\pi(S)$ be the minimal period of the sequence S if such a period exists and otherwise $\pi(S) = n$.*

Lemma 7.10. *For any sequence S with period $\pi(S) = 2$, it holds that $M(S) = 7$.*

Proof. Assume w.l.o.g. that $S = \mathsf{ACAC}\ldots\mathsf{AC}$. The only labels that can be considered are the ones that are subsequences of S, i.e., the possible labels are

$$\bigcup_{t=0}^{\frac{n}{2}-1} \left\{ (\mathsf{AC})^{t+1}, (\mathsf{CA})^{t+1}, (\mathsf{AC})^t\mathsf{A}, (\mathsf{CA})^t\mathsf{C} \right\}$$

Furthermore, for any $1 \leq t \leq n/2$ we have that $\{\mathsf{AC}\} \equiv_S \{(\mathsf{AC})^t\}$, and $\{\mathsf{CA}\} \equiv_S \{(\mathsf{CA})^t\}$, and for any $1 \leq t \leq n/2 - 1$ we have that $\{\mathsf{ACA}\} \equiv_S \{(\mathsf{AC})^t\mathsf{A}\}$, and $\{\mathsf{CAC}\} \equiv_S \{(\mathsf{CA})^t\mathsf{C}\}$. Thus, it is sufficient to consider only the labels in $\{\mathsf{A}, \mathsf{C}, \mathsf{AC}, \mathsf{CA}, \mathsf{ACA}, \mathsf{CAC}\}$. It can be verified that the labeling of S by each of these labels is unique, and by considering the empty codeset we have an S-uniquely decodable code of size 7. To see that no larger code is S-uniquely

decodable, note that any possible codeset is S-equivalent to some codeset that is composed only of labels in $\{\emptyset, \mathsf{A}, \mathsf{C}, \mathsf{AC}, \mathsf{CA}, \mathsf{ACA}, \mathsf{CAC}\}$ and that any codeset that is a subset of the last six labels, is S-equivalent to a codeset that consists of a single label out of the latter six labels. □

The upper bound in Lemma 7.10 can be extended to any period π by similar arguments. However, for $\pi > 2$ this upper bound is not necessarily tight. This result is summarized in the following lemma while the proof is left for the full version of the paper.

Lemma 7.11. *For any sequence S, we have that*

$$M(S) \leq 2^{2\pi(S)-2} + 2^{\pi(S)} - 1.$$

7.3 Fixed-Length Labels

In this section, we consider the special case in which all the labels have the same length. That is, we consider Problem 7.2 in the special case where the executable labels are all the labels of length ℓ, for some integer ℓ, i.e., $\mathcal{V} = \Sigma^\ell$. To this end, for a reference sequence S and an integer n we define $M_\ell(S) \triangleq M(S, \Sigma^\ell)$ and $M_\ell(n) \triangleq M(n, \Sigma^\ell)$. Similarly, we use the notations $S_{(n,\ell)}$ and $\mathcal{C}_{(n,\ell)}$ for the reference sequence and code defined in Problem 7.2.2.

Definition 7.12. *A labeling code $\mathcal{C}$ is called an* ℓ-labeling code *if for any codeset $\Lambda \in \mathcal{C}$ and any $\lambda \in \Lambda$ it holds that $|\lambda| = \ell$.*

We start with an upper bound on $M_\ell(n)$ for any two integers $0 < \ell \leq n$. Then, we show that in the special case where $\ell \geq \log_4(n - \ell + 1)$, this bound is tight. Before we present the bound and an explicit construction that meets it for $\ell \geq \log_4(n - \ell + 1)$, let us define $\eta(n, \ell)$ to be the number of binary sequences of length n in which each run of consecutive *ones* is of length at least ℓ. Additionally, define the constraint $\mathcal{T}_\ell$ to be the set of all binary sequences in which any run of *ones* is of length at least ℓ. This constraint is strongly related to the well-known *run-length limited* (*RLL*) constraint, which is described in the next definition [16].

Definition 7.13. *A binary sequence satisfies the* (d, k)-RLL constraint *if between every two consecutive* ones*, there are at least d* zeros *and there is no run of* zeros *of length $k + 1$. Denote the set of all sequences of length n that satisfy the (d, k)-RLL constraint by $\mathcal{C}_{d,k}(n)$.*

We note that the constraint $\mathcal{T}_\ell$ is equivalent to the (ℓ, ∞)-RLL constraint (replacing the roles of zeros and ones). It has been proven that for any constant ℓ, $\mathsf{cap}(\mathcal{C}_{\ell,\infty}) = \log_2 \rho$ where ρ is the largest real root of $x^{\ell+1} - x^\ell - 1$ [17], which implies the following corollary.

Corollary 7.14. *Let ℓ be a positive integer. We have that $M_\ell(n) \leq 2^{\Theta(n \cdot \log_2 \rho)}$, where ρ is the largest real root of $x^{\ell+1} - x^\ell - 1$. Furthermore, for a constant ℓ, it holds that*

$$\lim_{n\to\infty} \frac{\log(M_\ell(n))}{n} \leq \log_2 \rho.$$

Theorem 7.15. *Let ℓ and n be positive integers. For any reference sequence $S \in \Sigma^n$, we have that $M_\ell(S) \leq \eta(n, \ell)$, and hence $M_\ell(n) \leq \eta(n, \ell)$.*

Proof. Let $\mathcal{C}$ be an ℓ-labeling code. Note that by the definition of ℓ-labeling code, for any $\Lambda \in \mathcal{C}$, we have that $\Lambda(S)$ is a binary string of length $|S|$, in which any run of *ones* is of length ℓ or more. Additionally, since $\mathcal{C}$ is S-uniquely-decodable, for any $\Lambda_1, \Lambda_2 \in \mathcal{C}$, we have that $\Lambda_1(S) \neq \Lambda_2(S)$, and thus, $|\mathcal{C}| \leq \eta(|S|, \ell)$. □

Next, we show that the bound in Theorem 7.15 is tight by presenting an explicit construction for a code $\mathcal{C}_{(n,\ell)}$ that meets this bound with equality when S is an *ℓ-repeat-free* sequence, i.e., any substring of length ℓ in S is unique [18]. Note that such a sequence S corresponds to a trail in the de Bruijn graph.

Construction 7.1. *Let S be an ℓ-repeat-free sequence of length n over Σ and let $\mathcal{T}_\ell(n) \triangleq \mathcal{T}_\ell \cap \{0,1\}^n$ be the set of all binary strings of length n, in which any run of* ones *is of length ℓ or more. For any $X = (x_1, \ldots, x_n) \in \mathcal{T}_\ell(n)$, we define the codeset Λ_X as follows.*

1) *Initialize an empty set $\Lambda_X = \varnothing$.*
2) *For any run of* ones *in X of length $\ell' \geq \ell$, $X_{[i;\ell']}$:*
   (a) *Add the labels $S_{[i;\ell]}, S_{[i+\ell;\ell]}, \ldots, S_{[i+\ell\cdot\lfloor \ell'/\ell \rfloor;\ell]}$ into Λ_X.*
   (b) *If ℓ'/ℓ is not an integer, add the label $S_{[i+\ell'-\ell;\ell]}$ into Λ_X.*

Finally, we define $\mathcal{C}_{(n,\ell)} = \{\Lambda_X : X \in \mathcal{T}_\ell(n)\}$.

Example 7.16. *Let $S \in \Sigma^{25}$, and $X \in \mathcal{T}_4(25)$ be the following sequences,*

$$S = \mathsf{AAAACAAAGAAATAACCAACGAACT},$$

$$X = 1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0.$$

The sequence S is a 4-repeat-free sequence, and the codeset Λ_X which is obtained by Construction 7.1 for the binary sequence X is $\Lambda_X = \{\mathsf{AAAA}, \mathsf{CAAA}, \mathsf{AGAA}, \mathsf{CAAC}\}$, where the first three labels correspond to the first run of ones *(in X) and the last label corresponds to the second run of* ones*.*

Theorem 7.17. *For any ℓ-repeat-free sequence $S \in \Sigma^n$, the code $\mathcal{C}_{(n,\ell)}$ obtained by Construction 7.1 is an S-uniquely-decodable ℓ-labeling code of size $\eta(n, \ell)$.*

Proof. For any $1 \leq i \leq n - \ell + 1$ the subsequence $S_{[i;\ell]}$ is of length ℓ by definition. Hence, for any X, Λ_X that is defined by the algorithm can only contain labels of length exactly ℓ. That is, $\mathcal{C}_{(n,\ell)}$ is an ℓ-labeling code. Furthermore, for any $X \in \mathcal{T}_\ell(n)$, we have that any label $\lambda \in \Lambda_X$ appears only once as a subsequence of S and hence $\Lambda_X(S) = X$. Thus, for any two distinct sequences $X, Y \in \mathcal{T}_\ell(n)$, we have that $\Lambda_X(S) = X \neq Y = \Lambda_Y(S)$ which implies that $\mathcal{C}_{(n,\ell)}$ is an S-uniquely-decodable code of size $\eta(|S|, \ell)$. □

We note that to encode a binary message, we first need to encode it into a sequence $X \in \mathcal{T}_\ell(n)$ and then apply Construction 7.1 on X. An efficient encoder of the $\mathcal{T}_\ell$ constraint can be achieved by constructing a deterministic finite automaton that accepts all the constrained words, and then utilizing the state-splitting algorithm [19] to design encoders with rate approaching the capacity.

The following lemma shows that the size of ℓ-labeling codes is maximized for S which is ℓ-repeat-free (if such S exists).

Lemma 7.18. *If $S \in \Sigma^n$ is not ℓ-repeat-free, then*

$$M_\ell(S) < \eta(n,\ell).$$

The proof of Lemma 7.18 is based on the fact that for any S which is not ℓ-repeat-free there exists a sequence in $\mathcal{T}_\ell(|S|)$ which cannot be obtained as a labeling of S. The detailed proof of Lemma 7.18 is left for the full version of the paper.

Let n and ℓ be two positive integers and let $S \in \Sigma^n$. Theorem 7.15 and Lemma 7.18 state that if there exists an ℓ-repeat-free sequence of length n, then an S-uniquely-decodable ℓ-labeling code $\mathcal{C}$ is of maximum size if and only if S is ℓ-repeat-free. However, it should be noted that for any sequence S of length n, if S is ℓ-repeat-free then $|S| \leq 4^\ell + \ell - 1$. That is, Construction 7.1 is relevant only for $\ell \geq \log_4(|S| - \ell + 1)$.

Corollary 7.19. *If $\ell < \log_4(n-\ell+1)$ then $M_\ell(n) < \eta(n,\ell)$.*

Corollary 7.20. *Let $S \in \Sigma^n$ be a reference sequence and let $\ell \geq \log_4(n-\ell+1)$. We have that $M_\ell(S) = M_\ell(n)$ if and only if S is an ℓ-repeat-free sequence.*

Corollary 7.21.

$$M_\ell(n) = \eta(n,\ell) \text{ if and only if } \ell \geq \log_4(n-\ell+1).$$

In the following lemma and corollary the value $\eta(n,\ell)$ is analyzed.

Lemma 7.22. *For any two positive integers n, ℓ, it holds that*

$$\eta(n,\ell) = \sum_{t=0}^{\lfloor \frac{n+1}{\ell+1} \rfloor} \binom{n-t(\ell-1)+1}{2t}.$$

Proof. Let A be the set of binary strings of length n in which each run of *ones* is of length at least ℓ. We start by partitioning A into two disjoint sets,

$$A^0 = \{(x_1,\ldots,x_n) \in A : \ x_1 = 0\}$$
$$A^1 = \{(x_1,\ldots,x_n) \in A : \ x_1 = 1\}.$$

Clearly, $\eta(n,\ell) = |A| = |A^0 \cup A^1| = |A^0| + |A^1|$ since $A^0 \cap A^1 = \emptyset$. Next, let us calculate $|A^0|$, the analysis for $|A^1|$ is done similarly. We have that $|A^0| = \sum_{r=1}^{n} |A^0_r|$, where A^0_r is the set of all sequences in A^0 that have exactly r runs (of zeros/ones). Since the first run of any sequence in A^0 is a run of *zeros*, any sequence in A^0_r contains $r_1 = \lceil (r-1)/2 \rceil$ runs of *ones* and $r_0 = r - r_1 = \lfloor (r+1)/2 \rfloor$ runs of *zeros*. Additionally, each run of *zeros* must contain at least a single *zero* and each run of *ones* must contain at least ℓ *ones*. Let $\left(\!\binom{m}{b}\!\right)$ denote the number of b-element combinations of m objects, with repetitions. We have that

$$|A^0_r| = \left(\!\binom{r}{n-\ell r_1 - r_0}\!\right) = \left(\!\binom{r}{n-\lceil (r-1)/2 \rceil(\ell-1) - r}\!\right)$$
$$= \binom{n-\lceil (r-1)/2 \rceil(\ell-1)-1}{r-1}$$

and

$$|A^0| = \sum_{r=1}^{n} \binom{n - \lceil (r-1)/2 \rceil (\ell-1) - 1}{r-1}.$$

Similarly, we have that

$$\begin{aligned} |A^1| &= \sum_{r=1}^{n} \left(\binom{r}{n - \lfloor (r+1)/2 \rfloor (\ell-1) - r} \right) \\ &= \sum_{r=1}^{n} \binom{n - \lfloor (r+1)/2 \rfloor (\ell-1) - 1}{r-1} \end{aligned}$$

and thus

$$\begin{aligned} \eta(n,\ell) = &\sum_{r=1}^{n} \binom{n - \lceil \frac{r-1}{2} \rceil (\ell-1) - 1}{r-1} \\ &+ \sum_{r=1}^{n} \binom{n - \lfloor \frac{r+1}{2} \rfloor (\ell-1) - 1}{r-1}. \end{aligned}$$

By utilizing the identity $\binom{n-1}{k-1} + \binom{n-1}{k} = \binom{n}{k}$, and by considering even and odd values of n separately, it can be shown that the latter expression is equal to

$$\eta(n,\ell) = \sum_{t=0}^{\lfloor \frac{n+1}{\ell+1} \rfloor} \binom{n - t(\ell-1) + 1}{2t}.$$

□

Recall that the upper bound in Corollary 7.14 is evaluated under the assumption that ℓ is a constant. Since Construction 7.1 requires that the value of ℓ will grow as a function of n, the latter capacity does not represent the achievable rates of our construction. Hence, in the next theorem, we present the asymptotic behavior of $\eta(n,\ell)$ for the case $\ell = c\log_4(n)$, for $c \geq 1$. The proof follows from analyzing the expression given in Lemma 7.22 for $\ell = c\log_4(n)$ when $n \to \infty$.

Theorem 7.23. *Let n be a positive integer and let $\ell = c\log_4(n)$ for $c \geq 1$. Additionally, let S be an ℓ-repeat-free sequence and let $\mathcal{C}_{(n,\ell)}$ be the code obtained by Construction 7.1. It holds that*

$$M_\ell(n) = M_\ell(S) = |\mathcal{C}_{(n,\ell)}| = \eta(n,\ell) = 2^{\Theta\left(\frac{\log\log(n)}{\log(n)} \cdot n\right)}.$$

To conclude this section, we discuss the case where we are interested in an ℓ-labeling code which is S-uniquely-decodable, while the sequence S is not ℓ-repeat free. We start by proving a sufficient condition for an ℓ-labeling code to be S-uniquely-decodable for a given sequence S (not necessarily ℓ-repeat-free). To this end, for an ℓ-labeling code $\mathcal{C}$ we define $\mathcal{C}_{(S)} \triangleq \{\Lambda \cap W_\ell(S) : \Lambda \in \mathcal{C}\}$.

Lemma 7.24. *Let $0 < \ell \leq n$ be two integers and let $S \in \Sigma^n$. Additionally let $\mathcal{C}$ be an ℓ-labeling code. Then, $\mathcal{C}$ is S-uniquely-decodable if the following two conditions hold:*

1) $|\mathcal{C}| = |\mathcal{C}_{(S)}|$.

2) For any $\Lambda \in \mathcal{C}$ *we have that no proper prefix of a label* $\lambda \in \Lambda$ *is a proper suffix of a label* $\lambda' \in \Lambda$.

The first condition in Lemma 7.24 implies that all the codesets in $\mathcal{C}$ are different when ignoring labels that do not appear as substrings of S. The second condition is that any codeset in $\mathcal{C}$ is a *non-overlapping code* [20–23], that is, for any codeset $\Lambda \in \mathcal{C}$ we have that no proper prefix of a label $\lambda \in \Lambda$ is a suffix of a label $\lambda' \in \Lambda$. In fact, this condition is too strong and we only need the latter to hold concerning the *embeddings* of the labels in S. Hence, we prove the following weaker sufficient condition instead.

Theorem 7.25. *Let* $S \in \Sigma^n$ *and let* $\mathcal{C}$ *be an* ℓ*-labeling code. Then,* $\mathcal{C}$ *is* S*-uniquely-decodable if the following two conditions hold:*

1) $|\mathcal{C}| = |\mathcal{C}_{(S)}|$.

2) For any $\Lambda \in \mathcal{C}_{(S)}$ *and any* $\lambda, \lambda' \in \Lambda$*, if there exists a sequence* $\boldsymbol{w} \in \Sigma^*$ *such that* λ *is a prefix of* $\boldsymbol{w}$*,* λ' *is a suffix of* $\boldsymbol{w}$*, and* $|\boldsymbol{w}| < 2\ell$*, then* $\boldsymbol{w}$ *is not a substring of* S.

Proof. Let $\mathcal{C}$ be an ℓ-labeling code that satisfies the two conditions in the claim. We first note that $|\mathcal{C}| = |\mathcal{C}_{(S)}|$ implies that for any two codesets in $\mathcal{C}$, their subsets which contain only the ℓ-substrings of S are distinct. Hence we can ignore the labels that are not substrings of S and prove that the second condition guarantees that for any $\Lambda \in \mathcal{C}$, the codeset Λ can be uniquely determined from $\Lambda(S)$ and S.

For any $\Lambda \in \mathcal{C}$, the second condition implies that the sequence $\Lambda(S)$ is a binary sequence of length n in which any run of *ones* is of length $j\ell$ for some integer $j \geq 1$. Furthermore, for any run of *ones* in $\Lambda(S)$ there is a unique way to partition it to non-overlapping segments of length ℓ. Thus, for any run of $j\ell$ *ones* in $\Lambda(S)$, $j \geq 1$, we have a unique set of non-overlapping labels of length ℓ that can create this run. That is, Λ can be uniquely determined given S and $\Lambda(S)$, which completes the proof. □

Assume $\ell < \log_4(n - \ell + 1)$, using Lemma 7.24, we can construct an ℓ-labeling code which is S-uniquely decodable for a reference sequence $S \in \Sigma^n$ as follows.

Construction 7.2. *Given* $\ell < \log_4(n - \ell + 1)$ *and a sequence* $S \in \Sigma^n$*, construct an* ℓ*-labeling code* $\mathcal{C}$ *as follows.*

1) Let $\mathcal{N}_\ell$ *be the non-overlapping code of length* ℓ *over* Σ *given in [23].*

2) Let $\mathcal{N}_{(\ell,S)} \triangleq \mathcal{N}_\ell \cap W_\ell(S)$.

3) Let $\mathcal{C} \triangleq \mathcal{P}(\mathcal{N}_{(\ell,S)})$*, where* $\mathcal{P}(\mathcal{N}_{(\ell,S)})$ *is the power set of* $\mathcal{N}_{(\ell,S)}$.

It was proven in [23], that $|\mathcal{N}_\ell| \geq 63 \cdot \frac{4^{\ell-5}}{\ell}$, which implies the following lower bound on $M_\ell(S)$ and $M_\ell(n)$.

Claim 7.26. *For any* $S \in \Sigma^n$ *we have that* $M_\ell(S) \geq 2^{|\mathcal{N}_{(\ell,S)}|}$*. In particular,* $M_\ell(n) \geq 2^{63 \cdot \frac{4^{\ell-5}}{\ell}}$.

Proof. Let S be a sequence of length n such that the prefix of length $4^\ell + \ell - 1$ of S is a de Bruijn sequence. In this case, $\mathcal{N}_{(\ell,S)} = \mathcal{N}_\ell$ and hence the size of the code $\mathcal{C}$ from Construction 7.2 is $|\mathcal{C}| = 2^{|\mathcal{N}_{(\ell,S)}|} \geq 2^{63 \cdot \frac{4^\ell - 5}{\ell}}$. Hence, $M_\ell(S) \geq 2^{63 \cdot \frac{4^\ell - 5}{\ell}}$ which implies the claim of the theorem. □

While Construction 7.2 and the bound in Claim 7.26 can be used for any sequence S, in case we can use any reference sequence of length n, these results can be improved, as shown in the next theorem.

Theorem 7.27. *For integers n and ℓ, if $\ell < \log_4(n - \ell + 1)$ then*

$$M_\ell(n) \geq \eta(4^\ell + \ell - 1, \ell) = 2^{\Theta\left(\frac{\log(\ell)}{\ell} \cdot (4^\ell + \ell - 1)\right)}$$

Proof. To see that the claim in the theorem holds, let S be a sequence of length n such that the prefix of length $4^\ell + \ell - 1$ of S is a de Bruijn sequence and the code $\mathcal{C}_{4^\ell+\ell-1,\ell}$ obtained by Construction 7.1 where the decoding is done using the de Bruijn prefix of S. □

7.4 Conclusions

This study presents a theoretical framework for utilizing DNA molecules to encode information, intending to circumvent expensive DNA synthesis and facilitate a more practicable approach to certain applications. Our proposed methodology suggests to leverage established biochemical techniques commonly employed in medical and biological research, such as CRISPR-Cas9 and gRNA reagents for labeling. Through comprehensive exploration, we establish upper bounds on achievable codes under specific conditions and introduce an efficient encoder-decoder pair optimized for maximal code size.

To make further progress, future research should prioritize adapting the model, constructions, and bounds to a more realistic scenario that incorporates noise. One form of noise to consider is the potential for a small number of labels to be missing. This can result from labeling failures at specific sites or false negatives during the identification of labeled locations. Additionally, it's essential to address synchronization errors that may arise when determining labels' positions.

Bibliography

[1] A. Moter and U. B. Göbel, "Fluorescence in situ hybridization (FISH) for direct visualization of microorganisms," *Journal of Microbiological Methods*, vol. 41, no. 2, pp. 85–112, 2000.

[2] B. Chen, W. Zou, H. Xu, Y. Liang, and B. Huang, "Efficient labeling and imaging of protein-coding genes in living cells using CRISPR-tag," *Nature communications*, vol. 9, no. 1, p. 5065, 2018.

[3] J. Jeffet, S. Margalit, Y. Michaeli, and Y. Ebenstein, "Single-molecule optical genome mapping in nanochannels: Multidisciplinarity at the nanoscale," *Essays in Biochemistry*, vol. 65, no. 1, pp. 51–66, 2021.

[4] H. Ma, A. Naseri, P. Reyes-Gutierrez, S. A. Wolfe, S. Zhang, and T. Pederson, "Multicolor CRISPR labeling of chromosomal loci in human cells," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 112, no. 10, pp. 3002–3007, 2015.

[5] J. Deen, C. Vranken, V. Leen, R. K. Neely, K. P. F. Janssen, and J. Hofkens, "Methyltransferase-directed labeling of biomolecules and its applications," *Angewandte Chemie International Edition*, vol. 56, no. 19, pp. 5182–5200, 2017.

[6] A. P. Young, D. J. Jackson, and R. C. Wyeth, "A technical review and guide to RNA fluorescence in situ hybridization," *PeerJ*, vol. 8, p. e8806, 2020.

[7] S. K. Tabatabaei et al., "DNA punch cards for storing data on native DNA sequences via enzymatic nicking," *Nature communications*, vol. 11, no. 1, p. 1742, 2020.

[8] A. Sadremomtaz, R. F. Glass, J. E. Guerrero, D. R. LaJeunesse, E. A. Josephs, and R. Zadegan, "Digital data storage on DNA tape using CRISPR base editors," *Nature Communications*, vol. 14, no. 1, p. 6472, 2023.

[9] M. Jinek, K. Chylinski, I. Fonfara, M. Hauer, J. A. Doudna, and E. Charpentier, "A programmable dual-RNA–guided DNA endonuclease in adaptive bacterial immunity," *Science*, vol. 337, no. 6096, pp. 816–821, 2012.

[10] L. Cong et al., "Multiplex genome engineering using CRISPR/CAS systems," *Science*, vol. 339, no. 6121, pp. 819–823, 2013,

[11] I. Amit et al., "CRISPECTOR provides accurate estimation of genome editing translocation and off-target activity from comparative NGS data," *Nature Communications*, vol. 12, no. 1, p. 3042, 2021.

[12] D. Hanania, D. Bar-Lev, Y. Nogin, Y. Shechtman, and E. Yaakobi, "On the capacity of DNA labeling," in *2023 IEEE International Symposium on Information Theory (ISIT)*, 2023, pp. 567–572.

[13] Y. Nogin et al., "Design of optimal labeling patterns for optical genome mapping via information theory," *Bioinformatics*, vol. 39, no. 10, p. btad601, 2023.

[14] M. Levy-Sakin and Y. Ebenstein, "Beyond sequencing: Optical mapping of DNA in the age of nanotechnology and nanoscopy," *Current Opinion in Biotechnology*, vol. 24, no. 4, pp. 690–698, 2013.

[15] V. Müller and F. Westerlund, "Optical DNA mapping in nanofluidic devices: Principles and applications," *Lab on a Chip*, vol. 17, no. 4, pp. 579–590, 2017.

[16] B. H. Marcus, R. M. Roth, and P. H. Siegel, "An introduction to coding for constrained systems," San Diego, CA, USA:Lecture Notes, 2001. [Online]. Available: `http://www.math.ubc.ca/~marcus/Handbook/`

[17] S. W. McLaughlin, N. J. Luo, and N. Q. Xie, "On the capacity of M-ary runlength-limited codes," *IEEE Transactions on Information Theory*, vol. 41, no. 5, pp. 1508–1511, 1995.

[18] O. Elishco, R. Gabrys, E. Yaakobi, and M. Médard, "Repeat-free codes," *IEEE Transactions on Information Theory*, vol. 67, no. 9, pp. 5749–5764, 2021.

[19] R. Adler, D. Coppersmith, and M. Hassner, "Algorithms for sliding block codes-an application of symbolic dynamics to information theory," *IEEE Transactions on Information Theory*, vol. 29, no. 1, pp. 5–22, 1983.

[20] M. Levy and E. Yaakobi, "Mutually uncorrelated codes for DNA storage," *IIEEE Transactions on Information Theory*, vol. 65, no. 6, pp. 3671–3691, 2019.

[21] V. I. Levenshtein, "Maximum number of words in codes without overlaps," *Problemy Peredachi Informatsii*, vol. 6, no. 4, pp. 88–90, 1970.

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

[23] S. R. Blackburn, "Non-overlapping codes," *IEEE Transactions on Information Theory*, vol. 61, no. 9, pp. 4890–4894, 2015.

Towards Practical DNA Storage Systems - Unpublished Papers

Chapter 8

Deep DNA Storage: Scalable and Robust DNA Storage via Coding Theory and Deep Learning

Daniella Bar-Lev, Itai Orr, Omer Sabary, Tuvi Eztion, and Eitan Yaakobi

Abstract

DNA-based storage is an emerging technology that enables digital information to be archived in DNA molecules. This method enjoys major advantages over magnetic and optical storage solutions such as exceptional information density, enhanced data durability, and negligible power consumption to maintain data integrity [1, 2]. To access the data, an information retrieval process is employed, where some of the main bottlenecks are the scalability and accuracy, which have a natural tradeoff between the two. Here we show a modular and holistic approach that combines Deep Neural Networks (DNN) trained on simulated data, Tensor-Product (TP) based Error-Correcting Codes (ECC), and a safety margin mechanism into a single coherent pipeline. We demonstrated our solution on 3.1MB of information using two different sequencing technologies. Our work improves upon the current leading solutions by up to x3200 increase in speed, 40% improvement in accuracy, and offers a code rate of 1.6 bits per base in a high noise regime. In a broader sense, our work shows a viable path to commercial DNA storage solutions hindered by current information retrieval processes.

8.1 DNA-Based Storage

There is an exponential growth in the global data sphere fueled by the proliferation of digital technologies such as artificial intelligence, the Internet of Things, widespread internet connectivity, and the growing number of interconnected devices. While the global data sphere is anticipated to reach 180 Zettabytes by 2025, current storage solutions are not expected to scale at nearly the same pace due to capacity limitations [3]. To address this urgent need of the dig-

ital age, researchers are turning to innovative solutions like DNA-based storage, recognizing its potential to revolutionize long-term data storage capabilities due to its extraordinary data density and durability [2].

A DNA molecule consists of four building blocks called nucleotides: Adenine (A), Cytosine (C), Guanine (G), and Thymine (T). A single DNA strand, also called oligonucleotide, is an ordered sequence of some combination of these nucleotides and can be abstracted as a string over the alphabet $\{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$. The ability to chemically synthesize almost any possible nucleotides sequence makes it possible to store digital data on DNA strands.

The standard in-vitro DNA-based storage pipeline consists of several steps and is shown in Figure 8.1.a. First, the binary data is encoded into sequences over the DNA 4-ary alphabet, which are referred to as encoded sequences. Next, the encoded sequences are synthesized by a DNA synthesizer. The DNA synthesizer produces multiple DNA strands (known as oligos) for each encoded sequence, as current synthesis technologies cannot produce one single strand per sequence. Moreover, the length of the strands produced by the synthesizer is typically bounded by roughly 200–300 nucleotides to sustain an acceptable error rate [4]. The synthesized strands are then stored in a storage container in an unordered manner. To access the data, a sample of the strands is taken from the storage container, amplified using Polymerase Chain Reaction (PCR), and then sequenced by a DNA sequencer. The sequencer processes the strands and generates reads, which are digital representations of the strands as sequences over the DNA alphabet. However, at this stage, the data is mixed in an unordered and random manner. Recovery of the original binary information is then obtained by a computational-heavy step, which is referred to as DNA information retrieval.

DNA as a storage medium has several unique attributes that distinguish it from its widespread digital counterparts and should be considered in the design of the information retrieval pipeline. The first attribute is the inherent redundancy obtained by the synthesis and the sequencing processes, where each synthesized DNA strand has several copies. The second is that the strands are not ordered in memory and thus it is not possible to know the order in which they were stored. The third attribute is the unique noise characteristics from both the synthesis and the sequencing processes which introduce errors to the reads. These errors affect data integrity and are mostly of three types, insertion, deletion, and substitution of symbols, where the error rates and their characteristics depend on the synthesis and sequencing technologies [5].

The information retrieval pipeline, also shown in Figure 8.1.a, can be partitioned into several main stages: clustering, reconstruction, and decoding. First, a clustering algorithm is performed on the obtained reads. In this step, the unordered set of reads is partitioned into groups, where the goal is to partition the reads such that all the reads in each group originate from the same encoded sequence. Second, a reconstruction algorithm should be applied to each cluster to predict the encoded sequence. The use of a clustering algorithm and a reconstruction algorithm utilizes the inherent redundancy of DNA synthesis and sequencing to correct most of the errors, but usually not all.

In the DNA reconstruction problem [6–8], the goal is to predict an encoded sequence from a set of reads. One of the challenges in the DNA-based storage channel is that we do not necessarily have control over the size of a cluster, and it is likely that this size is significantly smaller than the required minimum size that guarantees a successful reconstruction of the encoded sequence according to the classical Levenshtein reconstruction problem [9]. Lastly, the decoding

step converts the reconstructed encoded sequence back to the binary data. In this step, if an ECC was applied during the encoding phase (prior to the DNA synthesis step), the remaining errors can be corrected using the decoding procedure of the ECC.

The first large-scale experiments that demonstrated the potential of in vitro DNA storage were reported by Church et al. [10] who stored 643KB of data and Goldman et al. [11] who accomplished the same task for a 739KB message. However, neither group recovered the entire message successfully. Later, Grass et al. [12] used a Reed Solomon (RS) based coding [13] solution in their DNA-based storage experiment, where they stored and recovered successfully an 81KB message. Erlich and Zielinski presented DNA Fountain [14], a coding scheme that is based on Luby transform. In their experiment, they stored and recovered 2.11MB of data. Organick et al. [15] developed and demonstrated a scheme that allows random access using DNA strands, where they stored 200MB of data. Yazdi et al. [16] presented a new coding scheme that was designed to correct errors from Nanopore sequencers, a smaller and portable sequencing technology that allows longer strands but has higher error rates. In their experiment they encoded 3.633KB of data which was successfully recovered. Wang et al. [17] stored 379.1KB using their suggested inner-outer scheme that combines cyclic redundancy check and repeat accumulate code while approaching an information capacity of 1.69 bits per base. Chandak et al. [18] suggested a coding method based on convolutional code and Recurrent Neural Network that integrates with Nanopore MinION. They were able to demonstrate their method and stored 11KB of information. Anavy et al. [19] demonstrated how the capacity of the DNA-based storage can be increased using composite letters.

The design of an information retrieval pipeline for DNA-based storage is a challenging problem as the errors include deletions and insertions. These errors are challenging types of errors, and many of the related theoretical problems are far from being solved [20]. This fact makes the design of both clustering and reconstruction algorithms more complex [21, 22]. Additionally, the clustering problem is a computationally intensive problem by itself. This is especially challenging when applied to DNA-based storage where the number of clusters can be extremely large, for example, 1TB of data will require an order of billions of clusters. Furthermore, theoretical reconstruction algorithms designed to address deletion and insertion errors, usually assumed that the clusters were partitioned (almost) perfectly [6, 23, 24] or were designed to work on a large block-length [23, 25–28].

Several works [12, 14, 18] tackled these difficulties by adding redundant symbols to each designed DNA strand (i.e., inner coding), or by adding redundant DNA strands (i.e., outer coding) to detect and correct the deletion and insertion errors. In these techniques, the clustering and the reconstruction steps can be avoided. In this approach, the inherent redundancy of the synthesis and the sequencing processes is not fully utilized which leads to suboptimal use of redundancy in the design of the ECC. This in turn can also lead to performance degradation due to an increase in the number of strands (and therefore also reads) that represent the information and should be processed.

Using machine learning methods for DNA-based storage was examined in Bee et al. [29], where the authors proposed a content-based similarity search and demonstrated how it can be added to DNA-based storage systems. In Pan et al. [30], the authors showed how data can be stored in DNA both in the strand and in its backbone structure, to create a rewriteable DNA-based storage system. Since these methods rely on the low entropy of the data, they are mainly

suitable in cases where there is a structured pattern in the data that the reconstruction method can exploit.

8.2 End-to-End Solution for DNA Information Retrieval

In this work, we present an end-to-end, practical solution to the in-vitro DNA information retrieval problem, as shown in Figure 8.1.a. Our solution, termed DNAformer, utilizes a modular encoding scheme, combining ECC and constrained codes prior to DNA synthesis and storage. Our coding scheme enables the pragmatic partitioning of large datasets into smaller blocks to allow for fast and easy access to specific parts within the data. When the information requires access, a sequencing process is used followed by an information retrieval process. The first step in this process is to bin the different reads into groups based on their index. This naïve approach introduces noise into the clusters which we treat in the following steps. The benefit is a significant increase in the speed of the clustering step over alternatives, more noise-tolerant approaches, such as the hash-based approach [22] and the Clover clustering approach [21].

In the second step, we utilize a DNN to reconstruct a sequence over the DNA alphabet based on a non-fixed number of reads with varying lengths and solve a sequence-to-sequence, Multiple-In-Single-Out problem. The model architecture shown in Figure 8.1.b uses a combination of convolutions and transformers followed by a confidence filter to screen correct predictions from false ones. The suspected incorrect predictions can go through a second reconstruction step, a dynamic programming-based algorithm, termed Conditional Probability Logic (CPL). The CPL algorithm analyzes the reads in each cluster and estimates their possible errors and probabilities according to the reads' similarity in the edit distance metric. The main advantage of the CPL algorithm is that it requires zero prior-knowledge of the cluster's error probabilities, and by estimating them it is possible to predict a good estimation even for small and erroneous clusters. This step adds another degree of freedom in our solution to deal with the tradeoff between accuracy and speed where the purpose is to deal with more challenging cases while not sacrificing the run-time capabilities of the system. Furthermore, our approach uses the concept of safety margin to quantify how robust the information retrieval pipeline is under specific working conditions.

The third step is the decoding of the data, where we utilize a modified TP code [31]. In our solution we modify the standard TP approach to take advantage of the first two steps, the clustering and reconstruction steps, and create a coherent, unified pipeline. This approach allows us to take advantage of the inherent redundancy and success of the upstream steps to reduce the redundancy within the coding scheme. Our scheme allows simple integration with a large family of constrained codes and flexibility with error correction capabilities.

Due to the high cost of acquiring large amounts of real data sufficient for training a DNN, we based our approach on simulated data. Our training methodology uses only a small amount of real data to model the error rates during the synthesis, PCR, and sequencing processes, and is done using the SOLQC tool [32]. Once these errors were modeled, simulated data can be generated to train a DNN in any quantity required. An important distinguishing property of this methodology is that the errors need to be modeled only once for each synthesis and sequencing processes, which makes our method scalable and cost effective.

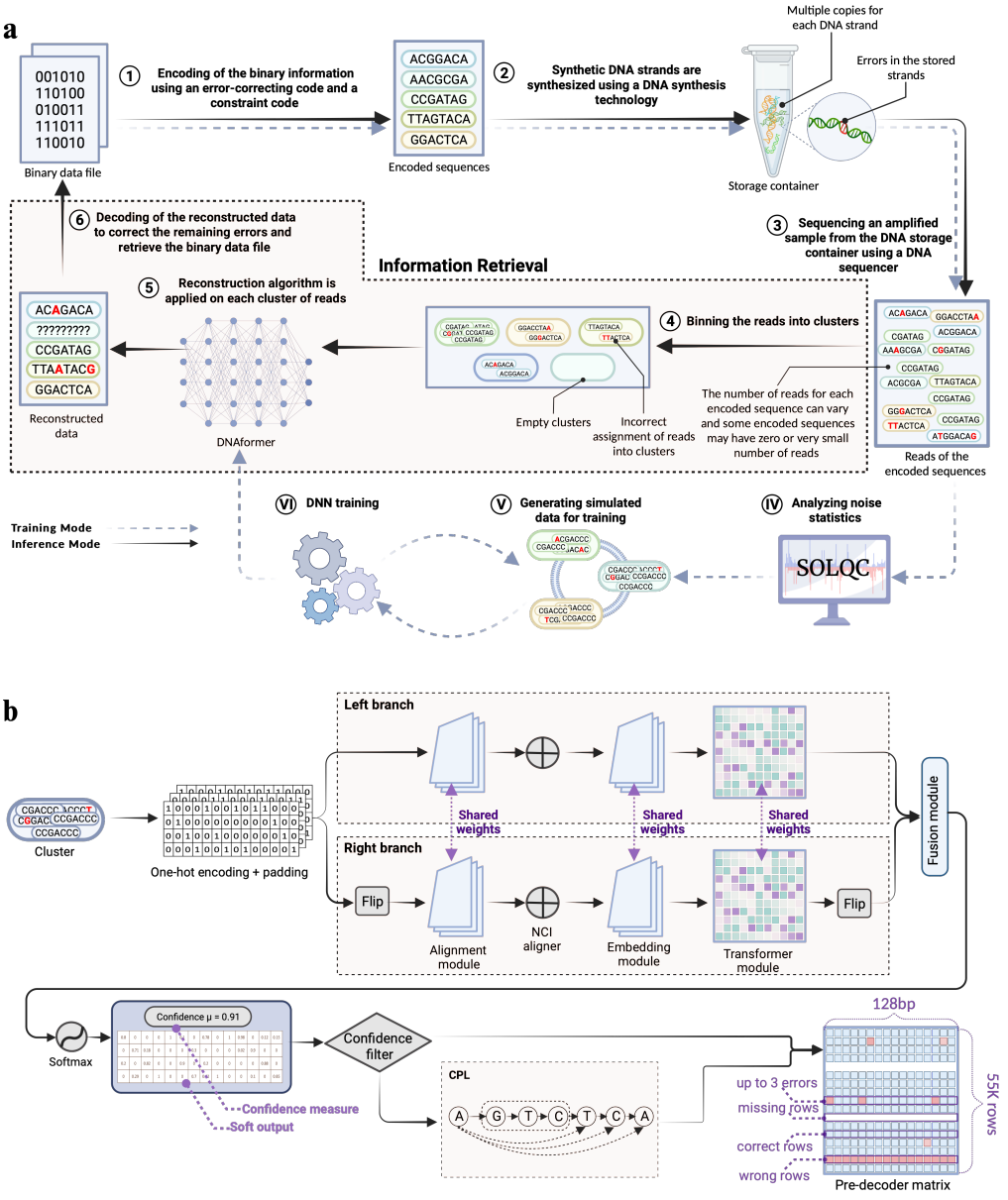


Figure 8.1: End-to-end solution for DNA information retrieval. a, shows a schematic description of our solution for the DNA-based storage pipeline. The numbers 1-6 depict the different stages through the process. In Roman numerals, we depict steps that are part of the training pipeline. b, shows a detailed view of the information retrieval process showing the DNN architecture, confidence filer, CPL, and the input to the decoder. This figure was created with BioRender.com.

A key point of our solution strategy is that we do not teach the model to utilize underlying semantics and file structure, but rather focus on the noise characteristics of the synthesis and sequencing processes. This gives the model the important ability to process unstructured and structured data with similar performance. Further details are provided in the Methods section and Supplementary Information.

8.3 DNA Dataset

Based our methodology, we partitioned the dataset into two parts. The first is a pilot dataset containing random information encoded into $1,000$ sequences to analyze the noise characteristic, and the second is the main dataset of $110,000$ encoded sequences, termed test dataset. The two datasets were synthesized by Twist Bioscience and obtained at different dates for both the synthesis and sequencing steps to make sure that they are as independent as possible.

The test dataset was composed of $110,000$ encoded sequences containing data from several sources: image, audio, text, and random information which was done to examine our approach on several different data modalities. The test dataset was purposefully split into two files, each containing $55,000$ encoded sequences. The first is a compressed (zipped) folder with three modalities: an image, a 24-second audio snippet, taken from Neil Armstrong's iconic moon landing, and a text file from the DNA Data Storage Alliance [33]. In total, the size of this compressed file is about 1.5MB. The second file contained about 1.5MB of random information bits. The data is shown in Figure 8.2.a-d. This partition was made to allow for the examination of random data in parallel to structured data.

Sequencing was done using two methods, Illumina miSeq and Oxford Nanopore MinION. This was done to examine our approach on different sequencing technologies each with its own pros and cons. Prior to each sequencing, we performed PCR amplification using the standard protocol of Q5 enzyme for 12 cycles [34]. Sequencing with Illumina miSeq produced $528,636$ reads for the pilot dataset, with an error rate of 0.079% and an average cluster size of 529. The test dataset had $3,215,249$ reads, 0.123% error rate, and an average cluster size of 29. The paired-end reads were merged using the PEAR software tool [35].

Sequencing with Oxford Nanopore MinION was done using the Ligation Sequencing Kit LSK 110 with flowcell 9.4.1 [36]. The pilot dataset consists of $2,805,705$ reads, with an error rate of 4.6% and an average cluster size of 754. The test dataset was sequenced twice, the first flowcell produced $4,341,575$ reads, an error rate of 4.1%, and an average cluster size of 13. The second flowcell produced $3,065,455$ reads, with an error rate of 5.07% and an average cluster size of 8. The combined two flowcells produced $7,407,030$ reads, an error rate of 4.47% and an average cluster size of 21. Two sequencing runs were performed to ensure the average cluster size was sufficiently large.

The noise characteristics of both the pilot and test datasets for both sequencing methods were obtained using the SOLQC tool [32] and are shown in Figure 8.2.e-f where we see a good fit between them. As our approach is based on generating simulated data for training, this illustrates the premise that it is possible to overcome challenges such as domain transfer between the simulated data used for training and the actual data during operation based on our approach. Additional details on our datasets can be found in the Supplementary Information.

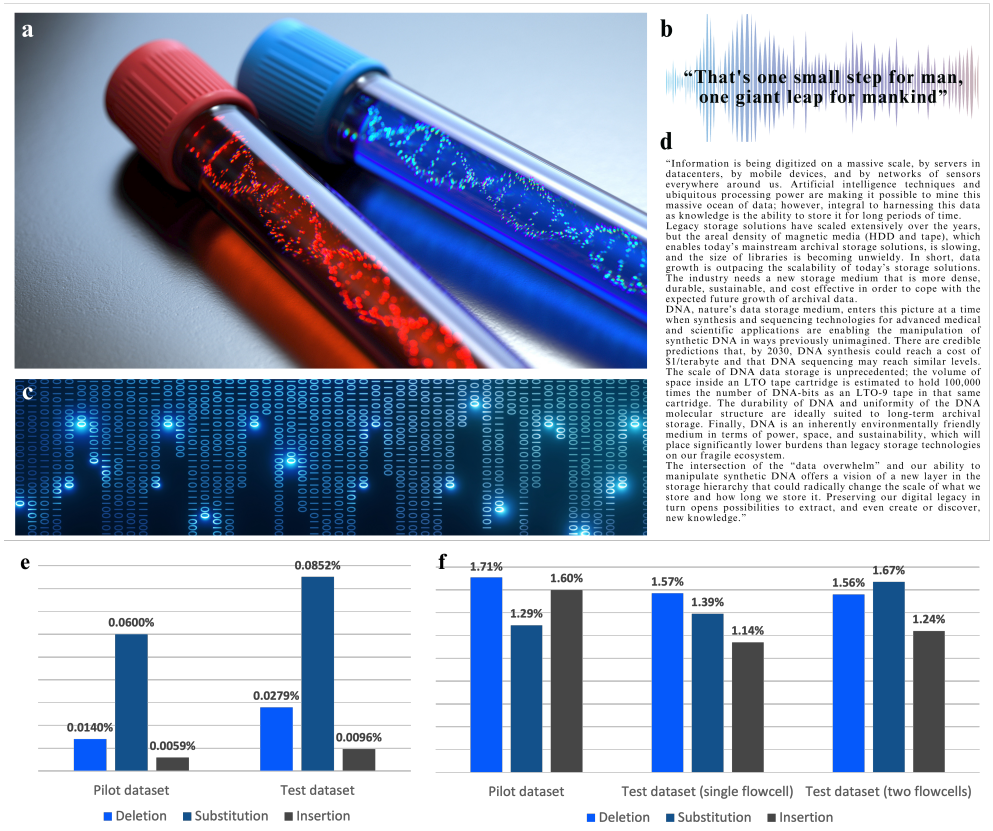


Figure 8.2: Data used for DNA experiments. a-d are an image, audio file, random bits and text accordingly. e, f, show statistical analysis of the errors found in the Illumina and Nanopore datasets accordingly. We deliberately chose these data modalities to examine the performance of our method under different conditions.

8.4 Results

To examine and compare the accuracy and performance of the DNAformer, we used both of our datasets, Illumina and Nanopore (termed after the sequencing technology used), as well as 4 additional publicly available datasets. The previously published datasets differ in their synthesis and sequencing technologies, leading to different noise characteristics. To allow a fair comparison of the different reconstruction algorithms on the datasets, all of them were clustered using our binning approach. Since some of the datasets do not include indices in their encoded sequences, we used the first unique symbols of these sequences as they were indices in the clustering process. The tested datasets were synthesized by Twist Bioscience, except the dataset by Grass et al. [12] which was synthesized by CustomArray. The sequencing technology for Grass et al. [12] and Erlich et al. [14] was Illumina miSeq, while Organick et al. [15] used Illumina NextSeq and Srinivasavaradhan et al. [7] used Oxford Nanopore MinION. Additional details on the publicly available datasets used can be found in the Supplementary Information.

For each dataset, we compared our method with 5 additional previously published algorithms. The first three algorithms use a symbol-wise majority vote approach in which the most frequent symbol in each position along the read is considered as the algorithm's prediction. These algorithms are based on the Bitwise Majority Alignment (BMA) [24] and are termed BMA Lookahead [37], Divider BMA [6], and VS algorithm [23]. The fourth method is the Iterative algorithm [6]. This algorithm uses dynamic programming methods to detect sequential patterns (small sub-sequences of the reads) that were observed in the reads and then uses the concatenation of the most frequent one as the algorithm's output. The fifth method is the hybrid algorithm [6], which uses the iterative algorithm and the Divider BMA algorithm, based on the given cluster and its estimated error probability.

Additionally, Srinivasavaradhan et al. [7] presented two trellis-based methods, the first is theoretical and the second is a heuristic improvement over the first, termed Trellis BMA. A comparison of the DNAformer with the Trellis BMA algorithm can be found in the Supplementary Information. A similar theoretical approach was also studied by Lenz et al. [8].

Notable work by Yazdi et al. [16] presented a coding scheme to correct errors caused by Nanopore sequencers and relied on coding constraints (such as exactly 50% of GC-content). As these constraints are not satisfied in our datasets, we did not include this method in our report.

A comparison of our approach to leading reported methods is provided in Figure 8.3.a, where the DNAformer achieves state-of-the-art (SOTA) results. On our Illumina and Nanopore datasets, the DNAformer achieves a failure rate of 0.0055% and 1.65% respectively. Additionally, our method achieves a failure rate of 0.02%, 0.66%, 0.17%, and 14.58% on the datasets provided by Erlich et al. [14], Grass et al. [12], Organick et al. [15], and Srinivasavaradhan et al. [7], respectively. We note that the closest method in reconstruction ability to the DNAformer is the iterative method [6].

We further compare the accuracy of the DNAformer on different data modalities in our dataset. The first file includes text, an audio message, and a photo, while the second is a file consisting of random bits. Our results, provided in the Supplementary Information, show similar accuracy for both files. Thus, showing our method does not rely on the underlying

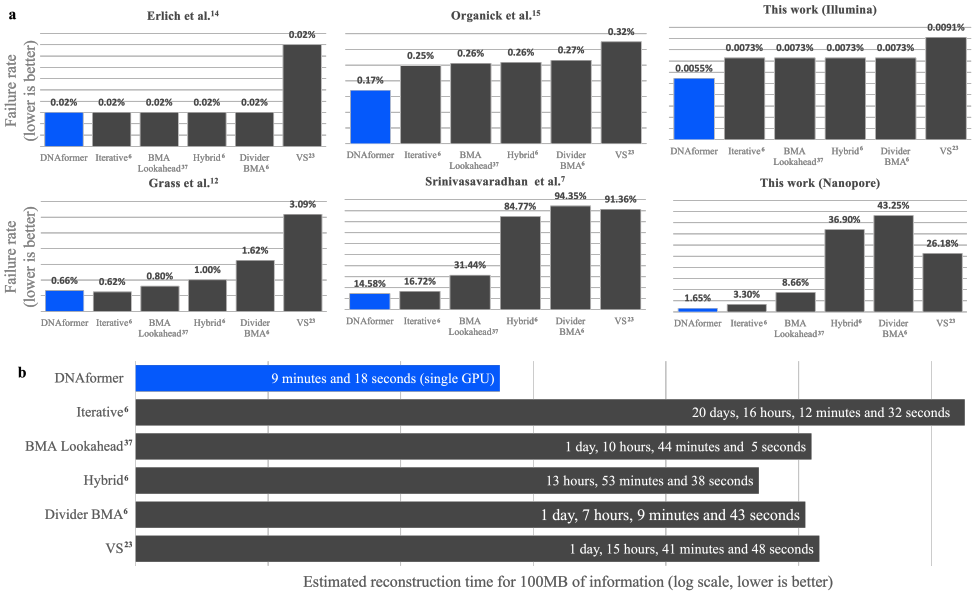


Figure 8.3: Comparison of the DNAformer to SOTA DNA reconstruction methods. a, shows the failure rate over several publicly available datasets as well as the Illumina and Nanopore test datasets, where the DNAformer achieves SOTA results. b, shows the estimated reconstruction for 100MB of information, where we see our method achieves a significant reduction in the required time.

semantics or structure in the data, but rather on the noise characteristics from the synthesis and sequencing processes. In addition to the reconstruction ability, it is also important to examine the inference speed of the different methods, as storage-based applications greatly favor fast processing speeds. The results are reported in Figure 8.3.b, where we compare the time, each method will take to reconstruct 100MB of data. Our method greatly improves on current methods by several orders of magnitude. When compared to the second-in-performance method (i.e. the Iterative method), we improve the speed by x3200. Meaning, the DNAformer simultaneously improves both reconstruction accuracy and speed without the natural tradeoff that usually exists between the two.

To guarantee the retrieval of binary data, our method uses a TP based, modular coding scheme. When comparing different coding schemes, an important parameter to consider is the code rate which signifies how efficient the code is in terms of redundant symbols. Furthermore, the code rate needs to be evaluated alongside the channel's error probability as an increase of this parameter makes the retrieval more challenging. Figure 8.4.a shows a comparison between different coding schemes where we see our work achieves a high code rate in addition to being utilized in a relatively high noise regime.

Apart from guaranteeing information retrieval the coding scheme presented in this work also introduces safety margins which signify how robust is the DNAformer under specific working conditions. As there are three main categories of errors from the DNAformer, where some are more difficult to correct than others, our code has been designed to support a range of

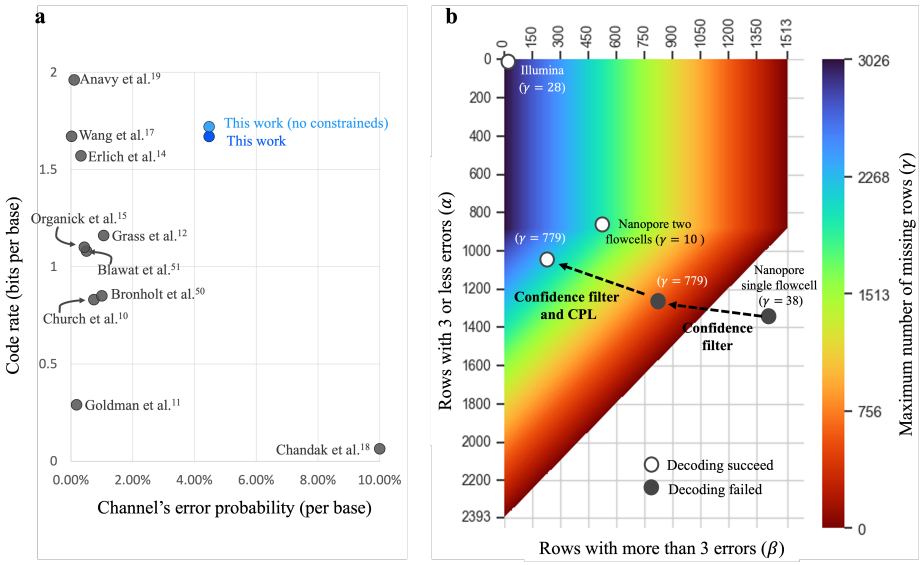


Figure 8.4: Evalution of information retrieval performance. a, comparison of the code rate versus the channel's error probability of different coding schemes that were used in DNA-based storage experiments. Our results show a competitive code rate even while operating in higher noise regimes. b, heatmap of the errors regime where our coding scheme is able to operate successfully. The specific instances of our three datasets are shown on top and illustrate a failure/success of the retrieval as well as their associated safety margin.

values for each error type. Figure 8.4.b shows a heatmap with the three types of errors the code can correct and α, β, γ denote the quantities of each type of error. We see that when tested on the Illumina dataset, the DNAformer is well within the safety margins for information retrieval. When tested on the Nanopore two flowcells, the DNAformer can retrieve the information, however, the safety margin has been decreased. When tested on the Nanopore single flowcell, we see that the DNAformer is not able at first to retrieve the information. Following the method shown in Figure 8.1, when applying both the confidence filter and the CPL, the DNAformer can successfully retrieve the information. Additional experimentation results, ablations studies, and comparisons are presented in the Supplementary Information.

8.5 Discussion

When considering commercial, real-life applications of DNA-based storage, several system-related considerations arise, such as robustness, scalability, run-time, and compute requirements. As this work presents an end-to-end solution to the DNA information retrieval problem, several individual components in the pipeline have been re-designed. Additionally, focus was placed on the interaction between the different components which is critical to a successful operation. The results show SOTA performance in accuracy and run-time speed on several publicly available datasets in addition to new datasets provided in this work. Extensive ex-

perimentation and ablation studies are provided in the Supplementary Information showcasing the modular nature of this approach and how it can be modified to different applications and settings.

Our solution is based on the combination of TP-based ECC wrapping a transformer-based DNN while considering the system-related considerations mentioned earlier. To overcome the runtime limitations of current DNA-based storage pipelines, while using the inherent redundancy of the DNA-based storage channel, our design uses a naïve and very efficient method for clustering. However, this comes at the cost of noise within each cluster. Therefore, the information retrieval pipeline needs to be able to overcome this type of noise.

DNNs are a good fit for these requirements due to their parallel computational nature and Graphics Processing Unit (GPU) implementations, which allowed us to achieve inference time several orders of magnitude faster than previous solutions. However, the current cost of producing a large volume of real data for training such a model is high. In addition, since there is a need to employ an ECC prior to the DNA synthesis process, some changes in the design of the code will require to generate a new dataset. For these reasons, we turn to simulated data to train our model.

During training, we utilized 1.4B simulated DNA reads, dwarfing any existing datasets for DNA-based storage. The cost to create such a large dataset from real DNA (i.e., not simulated data) is estimated at over $10M, far greater than most labs' budget. Showing how our method paves the way for a realistic, large-scale application of DNA-based storage. Our training methodology uses a small amount of data from real experiments to model the errors, from which we can create an unlimited amount of simulated training data.

The proposed coding scheme introduces different attributes. First, we consider the entire information retrieval pipeline, which allows a simplification of the channel. Our approach strips away complexities involved with correcting insertions and deletions, making the process simpler. Second, by strategically leveraging a TP code, instead of the inner-outer code approach, we demonstrated that the integration of the clustering and reconstruction steps into the decoding process leads to a significant reduction in the required number of redundant symbols for error correction. Third, we offer a modular way to incorporate various constrained codes into our scheme, a known challenge in Coding Theory [13, 38]. This allows for a versatile means of adapting to different conditions or requirements, enhancing the adaptability of our approach across various scenarios.

From a system-design perspective using the confidence filter and CPL algorithm increases the safety margin and allows our method to expand its ability to solve difficult cases. For example, higher noise regimes or smaller cluster sizes, a desired trait in terms of cost and time.

Our proposed method of integrating ECC and constrained codes extends outside the domain of DNA-based storage and can be useful in other channels as well. Moreover, the utilization of a TP based scheme that relies on similar ideas can be used for scenarios in which a DNN (or any other algorithm) can be applied to parts of the data pre-decoding.

8.6 Conclusion

In this work, we present a scalable method for DNA-based storage that overcomes some of the major bottlenecks in current solutions for balancing failure rate and run-time. Our method combines a DNN and ECC to leverage the sequencing and synthesis inherent redundancy without compromising on performance. Furthermore, our solution reduces the required ECC redundancy and the required number of reads for error-free recovery of the information. Our results showed that DNNs can significantly improve the decoding process in a DNA-based storage system and shorten a DNA-based storage system response time by several orders of magnitude. From a broader perspective, our DNN-ECC approach overcomes a major bottleneck on the path to large-scale commercial applications of DNA-based storage.

Bibliography

[1] L. C. Meiser et al., "Synthetic DNA applications in information technology" *Nature Communications*, vol. 13, no. 1, p. 352, 2022.

[2] L. Ceze, J. Nivala, and K. Strauss, "Molecular digital data storage using DNA." *Nature Reviews Genetics*, vol. 20, no. 8, pp 456–466, 2019.

[3] D. R. J. G. J. Rydning et al., "The Digitization of the World From Edge to Core," *Framingham: International Data Corporation* vol. 16, pp. 1–28, 2018.

[4] E. M. LeProust et al., "Synthesis of high-quality libraries of long (150mer) oligonucleotides by a novel depurination controlled process," *Nucleic Acids Research*, vol. 38, no. 8, pp. 2522–2540, 2010.

[5] R. Heckel, G. Mikutis, and R. N. Grass, "A Characterization of the DNA data storage channel," *Scientific Reports*, vol. 9, no. 1, p. 9663, 2019.

[6] O. Sabary, A. Yucovich, G. Shapira, and E. Yaakobi "Reconstruction algorithms for DNA storage systems," *Scientific Reports*, vol. 14, no. 1, p. 1951, 2024.

[7] S. R. Srinivasavaradhan, S. Gopi, H. D. Pfister and S. Yekhanin, "Trellis BMA: Coded Trace Reconstruction on IDS Channels for DNA Storage," in *2021 IEEE International Symposium on Information Theory (ISIT)*, 2021, pp. 2453–2458.

[8] A. Lenz, I. Maarouf, L. Welter, A. Wachter-Zeh, E. Rosnes, and AG i Amat, "Concatenated codes for recovery from multiple reads of DNA sequences," in *2020 IEEE Information Theory Workshop (ITW)*, 2021, pp.1–5.

[9] V. I. Levenshtein, "Efficient reconstruction of sequences from their subsequences or supersequences," *Journal of Combinatorial Theory. Series A*, vol. 93, no. 2, pp. 310–332, 2001.

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

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

[12] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angewandte Chemie International Edition*, vol. 54, no. 8, pp. 2552–2555, 2015.

[13] F. J. MacWilliams and N. J. A. Sloane, "The theory of error-correcting codes," *Elsevie*, vol. 16, 1997.

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

[15] L. Organick et al., "Random access in large-scale DNA data storage," *Nature Biotechnology*, vol. 36, no. 3, pp. 242–248, 2018.

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

[17] Y. Wang, et al. "High capacity DNA data storage with variable-length Oligonucleotides using repeat accumulate code and hybrid mapping," *Journal of Biological Engineering*, vol. 13, pp. 1–11, 2019.

[18] S. Chandak et al., "Overcoming high nanopore basecaller error rates for DNA storage via basecaller-decoder integration and convolutional codes," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 8822-8826.

[19] L. Anavy, I. Vaknin, O.Atar, R. Amit and Z. Yakhini, "Data storage in DNA with fewer synthesis cycles using composite DNA letters". *Nature Biotechnology,* vol. 37, no. 10, pp. 1229–1236, 2019.

[20] M. Cheraghchi and J. Ribeiro, "An overview of capacity results for synchronization channels," *IEEE Transactions on Information Theory*, vol. 67, no. 6, pp. 3207–3232, 2020.

[21] G. Qu, Z. Yan, and H. Wu, "Clover: tree structure-based efficient DNA clustering for DNA-based data storage," *Briefings in Bioinformatics*, vol. 23, no. 5, 2022.

[22] C. Rashtchian et al., "Clustering billions of reads for DNA data storage," *Advances in Neural Information Processing Systems*, vol. 30, pp. 3360–3371, 2017.

[23] K. Viswanathan and R. Swaminathan, "Improved string reconstruction over insertion-deletion channels," *Symposium on Discrete Algorithms* 2008, pp. 399–408.

[24] T. Batu, S. Kannan, S. Khanna, and A. McGregor, "Reconstructing strings from random traces," in *SODA* 2004, vol. 4, pp. 910–918.

[25] N. Holden, R. Pemantle, and Y. Peres, "Subpolynomial trace reconstruction for random strings and arbitrary deletion probability," *Conference On Learning Theory*, 2018, pp. 1799–1840.

[26] T. Holenstein, M Mitzenmacher, R. Panigrahy, and U. Wieder, "Trace reconstruction with constant deletion probability and related results," *Symposium on Discrete Algorithms*, 2008, pp. 389–398.

[27] F. Nazarov and Y. Peres, "Trace reconstruction with $\exp(O(n^{1/3}))$ samples," in *ACM SIGACT Symposium on Theory of Computing*, 2017, pp. 1042–1046.

[28] Y. Peres and A. Zhai, "Average-case reconstruction for the deletion channel: Subpolynomially many traces suffice," in *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, 2017, pp. 228–239.

[29] C. Bee, et al. "Molecular-level similarity search brings computing to DNA data storage," *Nature communications*, vol.12, p. 4764, 2021.

[30] C. Pan, S. K. Tabatabaei, S. M. H. T. Yazdi, A. G. Hernandez, C. M. Schroeder, and O. Milenkovic, ' "Rewritable two-dimensional DNA-based data storage with machine learning reconstruction," *Nature Communications*, vol. 13, no. 1, pp. 1–12, 2022.

[31] J. Wolf, "On codes derivable from the tensor product of check matrices," *IEEE Transactions on Information Theory*, vol. 11, no. 2, pp. 281–284, 1965.

[32] O. Sabary, Y. Orlev, R. Shafir, L. Anavy, E. Yaakobi, and Z. Yakhini "SOLQC: Synthetic oligo library quality control tool," *Bioinformatics*, vol. 37, no. 5, pp. 720–722, 2021.

[33] DNA data storage alliance, "Preserving our digital legacy: An introduction to DNA data storage," a publication of *DNA Data Storage Aliance*, 2021.

[34] J. F. Menin and N. M. Nichols, "Multiplex PCR using Q5® High-Fidelity DNA Polymerase," *N. Engl. Biolabs Appl. Note.*, 2013.

[35] J. Zhang, K. Kobert, T. Flouri, A. Stamatakis, "PEAR: A fast and accurate Illumina Paired-End reAd mergeR," *Bioinformatics*, vol. 30, no. 5, pp 614–620, 2014.

[36] Y. Wang, Y. Zhao, A. Bollas, Y. Wang, and K. F. Au, "Nanopore sequencing technology, bioinformatics and applications," *Nature Biotechnology*, vol. 39, no. 11, pp. 1348–1365, 2021.

[37] P. S. Gopalan, S. Yekhanin, S. D. Ang, N. Jojic, M. Miklos, K. Strauss, and L. Ceze, "Trace reconstruction from noisy polynucleotide sequencer reads," U.S. Patent Application No. 15/536,115, 2018.

[38] B. H. Marcus, R. M. Roth, and P. H. Siegel, "An introduction to coding for constrained systems," San Diego, CA, USA:Lecture Notes, 2001. [Online]. Available: `http://www.math.ubc.ca/~marcus/Handbook/`

[39] A. L. Gimpel, W. J. Stark, R. Heckel, R. N. Grass, ' "A digital twin for DNA data storage based on comprehensive quantification of errors and biases," *Nature Communications*, vol. 14, no. 1, p. 6026, 2023.

[40] J. Bohlin, B. Rose, J. H. O. Pettersson, "Estimation of AT and GC content distributions of nucleotide substitution rates in bacterial core genomes," *Big Data Analytics*, vol. 4, pp. 1–11, 2019.

[41] F. Weindel, A. L. Gimpel, R. N. Grass, and R. Heckel, "Embracing errors is more effective than avoiding them through constrained coding for DNA data storage," *Annual Allerton Conference on Communication, Control, and Computing*, 2023, pp 1–8.

[42] N. Stoler and A. Nekrutenko "Sequencing error profiles of Illumina sequencing instruments," *NAR Genomics and Bioinformatics*, vol. 3, no. 1, 2021.

[43] Z. Ping, et al. "Towards practical and robust DNA-based data archiving using the yin–yang codec system," *Nature Computational Science*, vol. 2, no. 4, pp. 234–242, 2022.

[44] G. Chaykin, N. Furman, O. Sabary, and E. Yaakobi, "DNA Storage Simulator," 2020.

[45] G. Chaykin, O. Sabary, N. Furman, D. Ben-Shabat, and E. Yaakobi, "DNA-storalator: end-to-end DNA storage simulator," *13th Annual Non-Volatile Memories Workshop* 2022.

[46] T. Xiao, M. Singh, E. Mintun, T. Darrell, P. Dollár, R. B. Girshick, "Early convolutions help transformers see better," *Advances in Neural Information Processing Systems*, vol. 34, pp. 30392–30400, 2021.

[47] B. Chowdhury and G. Garai, "A review on multiple sequence alignment from the perspective of genetic algorithm," *Genomics*, vol. 109, no. 5–6, pp. 419–431, 2017.

[48] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017 pp. 1251–1258.

[49] A. Vaswani, et al. "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[50] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "A DNA-based archival storage system," *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2016, pp 637–649.

[51] M. Blawat et al., "Forward error correction for DNA data storage," *Procedia Computer Science*, vol. 80, pp. 1011–1022, 2016.

[52] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *Journal of the ACM (JACM)*, vol. 21, no. 1, pp. 168–173, 1974.

8.7 Methods

Our method is an end-to-end solution to the DNA information retrieval problem, as shown in Figure 8.1.a. As part of this approach, it was divided into several components, each with its own functionality: encoding of sequences prior to DNA synthesis, clustering of reads post

sequencing, reconstruction and decoding back to the original data. In addition, the interplay and interface between the different components was also taken into consideration as part of a complete system-level, optimized solution. The solution combines coding theory and deep learning methods to create a holistic and coherent pipeline to encode and decode the DNA data.

One of the merits of DNA-based storage is high data density, meaning a scalable storage system needs to be able to quickly process arbitrary large files. To create a scalable method, we do not require the entire file to be processed simultaneously and design our method to process data in smaller batches, hence, our solution can be adapted to random access purposes. Our suggested solution encodes a block of 1.53962MB into 55k designed encoded sequences of length 140. Each strand is composed of 12 bases for index, including a single base that serves as a file identifier. The remainder 128 bases were allocated for the binary data and the redundancy from the ECC and constrained code. This length was chosen to allow for efficient DNN processing on a GPU during the reconstruction step and the coding scheme during the encoding and decoding steps.

8.7.1 Coding Scheme

Our coding scheme is a modular pipeline composed of several components, each addressing a different purpose: index encoding, diagonal column encoding, constrained code and TP code. The complete encoding and decoding pipeline is designed to integrate these four components in an interleaving order that allows each of them to achieve its designed goal without hindering the others.

There are three main considerations in the design of our coding scheme. First, our solution assumes that the decoding is performed after the clustering and reconstruction algorithms. By design, these steps eliminate most of the deletions and insertions errors, and the output has the same length as the encoded sequences. Hence, after this step, we need to only take care of substitution errors and missing predictions due to either missing clusters or our confidence filter, which is easier to solve in terms of coding theory and requires less redundancy.

As seen in Figure 8.3.a, when the clustering and reconstruction perform well, a high number of error-free predictions can be passed to the decoder, so the decoder only needs to correct errors in a small fraction of the predicted sequences, rather than in all of them. Therefore, a main component of our code is a TP based coding scheme that can correct up to a specified number of errors in up to a specified fraction of the predictions, allowing for a significant reduction in the redundancy needed to ensure error-free reconstruction of the data.

To maintain error-free retrieval of the information, our code also uses a diagonal column encoding which serves as an outer code that can correct the remaining erroneous predictions and overcome the missing ones. This code is implemented using an RS code which is applied on the encoded sequences that store the information. As some sequencing and synthesis technologies are more error-prone at the beginning and end of each read [32, 39] the RS code was designed diagonally. Meaning, the information at the beginning and end of each read will be encoded together with the information in the middle to create a more uniform distribution of the errors.

The second consideration in our design is that our clustering step is essentially a binning algorithm that uses the index part of the reads and matches them to the valid indices that were

used in the designed sequences. Hence having errors in the index can cause a read to be ignored or misclassified. As misclassified reads are more problematic during the reconstruction step, we use an index encoding that is based on a pre-calculated set of indices that are of edit distance 3 or more from each other.

Our third consideration relates to constrained coding for DNA-based storage. The different sequencing and synthesis technologies lead to different constraints that should be addressed, depending on the specific technology. For example, in our datasets, we selected the mapping to avoid the long homopolymer (consecutive repetition of the same base) of length 5 or more and GC-content of each encoded sequence between 45% to 55% with high probability. These constraints were selected to preserve the stability of the synthesized strands and mitigate their error rates, considering the technologies that are used in our experiments [40–42].

Additionally, recent work demonstrated that in some cases, it is better to not impose any constraint [41]. Hence, to keep our code flexible for different use cases we designed a block-based constrained code in a way that is almost independent of other components of our scheme. This allows an easy adaptation to other constraints or a removal of the constrained code component entirely. In our design, removing the constrained code improves the information rate by 7.6%.

8.7.2 Clustering

From a system-level optimization perspective, this step was optimized for speed rather than accuracy. We adopt a naïve clustering approach based on simple and fast binning of the reads based on their index. The goal in our design for this step is to overcome slow clustering methods [21, 22] and reduce the processing time. However, this comes at the cost of noise, introduced by errors in the index of each read, which causes some of the reads to be wrongly clustered (i.e. false reads) or to be dropped during the clustering step. On a system-level basis, we overcome this noise in the reconstruction step using a DNN.

8.7.3 Reconstruction

Designing a method that combines a DNN and ECC requires the ability to iterate between the two parts during the design phase. That is, the coding scheme and the DNN are coupled together to guarantee a specific set of success metrics. However, creating a different training dataset for each coding scheme modification is a costly and resource intensive process. For example, using previous DNA-based storage systems [14, 43], the estimated cost of synthesizing 1GB of data is roughly $3-5M.

Due to this fact, we turn to simulated data for training our DNN. The main challenge when using simulated data for training is the generalization to real-world settings after the model is trained. To overcome this issue, we construct a data generator based on statistics from real-world experiments [32, 44]. These statistics contain the error probabilities which are used to generate the reads of each label.

Simulated Data Generation

Our reconstruction approach uses a DNN which predicts a single label sequence from a cluster of reads. Since the data is randomly sampled, each cluster can vary in size, and each read can vary in length due to synthesis and sequencing errors. Moreover, some clusters can suffer from higher error rates compared to others. Our method utilized simulated data only during training, meaning, we did not use real data during the training of the models. Pseudo code for each iteration of the training process using our simulated data generator is given below:

1) Draw a random sequence of letters based on the 4-ary $\{\mathsf{ACGT}\}$.
2) Encode the sequence using some error-correcting and constrained coding scheme (optional).
3) Draw a random number of reads and a random number of false reads.
4) For each read:
   (a) Draw random deviation from the modeled error probabilities.
   (b) Inject simulated errors for each read based of the error model [45].
5) Batch several clusters.
6) Forward-backward pass through the DNN.

Data Preprocessing

The DNAformer architecture processes the reconstruction of a cluster of reads. Meaning, a set of reads is processed simultaneously to reconstruct the suitable encoded sequence. Preparation of the reads includes filtering short and long reads beyond a specified design parameter. This ensures that highly corrupted reads will not affect the reconstruction process. Following, a simple one-hot encoding and padding are used to prepare the data to the model's encoder and make sure all reads are of the same length.

Model Architecture

Our model uses a combination of convolutions and transformers. We adopt the concept of early convolutions before a transformer block to improve training stability and performace [46]. The model is structured as a Siamese network where the two branches share weights and are different by reversing the symbol order prior to the branch termed 'right branch' in Figure 8.1.b Fig 1b. This is done since the padding used in the preprocessing step is concatenated only at the end of each read. An alternative approach will be to align all the copies to one another using sequence alignment methods [47]. However, these methods come at the cost of 'in-series' computation and processing time and therefore we decided to avoid them.

Instead, we created an alignment module whose purpose is to learn the required alignment for each read independently. The alignment module uses an Xception [48] inspired architecture with depthwise separable convolutions and multiple kernel heads. The purpose of using

multiple kernels in the embedding layer is to allow the model to learn different shifts caused by deletion or insertion errors. Following this module, we sum over the cluster dimension with the goal of improving the model's robustness to differences in cluster size. This operator has the effect of Non-Coherent Integration (NCI) and aims to increase the Signal to Noise Ratio (SNR) of the data.

After the NCI aligner layer, the model includes an embedding module whose architecture is similar to the alignment module, however, now the operations are employed on the whole cluster and not on each read independently. The goal is to learn correlations between the different reads and prepare the data for the Transformer module. In addition, the embedding module outputs a sequence with the required output length and larger feature space.

The transformer module is a multi-head transformer architecture, used with Multi-Layer Perceptron as feedforward layers [49]. We do not use position embedding in this module. After the last transformer block, a linear module is used to reduce the number of features to 4 which represent one-hot encoding for the DNA representation.

The fusion layer is a vector of learnable parameters with a length of the required encoded sequence. This layer combines between the predictions of the two branches into a single prediction prior to a softmax operator which transforms this representation to probabilities. Additional ablation studies and details on the architecture are provided in the Supplementary Information.

Loss Function

To train our model a combination of cross entropy and consistency loss was used and are shown in Eq. (7.1), (7.2), and (7.3):

$$\mathcal{L} = \lambda_{\text{ce}}\mathcal{L}_{\text{ce}} + \lambda_{\text{consistency}}\mathcal{L}_{\text{consistency}} \tag{8.1}$$

$$\mathcal{L}_{\text{ce}} = -\frac{1}{n}\sum_{n} y_n \log(\Theta_n) \tag{8.2}$$

$$\mathcal{L}_{\text{consistency}} = 0.5 \cdot \left(\mathcal{L}_{\text{ce}}(\Theta_{\text{Left Branch}}) + \mathcal{L}_{\text{ce}(\Theta_{\text{Right Branch}})}\right) \tag{8.3}$$

Where λ_i are hyperparameters, Θ_i are the model prediction probabilities and y_n are the labels. Left and right branches refer to the Siamese architecture. The formulation for consistency loss which yielded best results is shown in Eq. (7.3) and aims to minimize the average prediction of the two branches, giving the model the freedom to adapt to changing noise characteristics along the sequence length.

Training Details

Data generation and training was implemented in Pytorch, optimizer used was Adam with $\beta_1 = 0.9, \beta_2 = 0.999$, batch size 64 and learning rate utilized cosine decay from $3.141 \cdot 10^{-5}$ to $3.141 \cdot 10^{-7}$. A single A40 GPU was used during training and inference. Training took 50 epochs for the Illumina experiment and 180 epochs for the Nanopore experiment, each containing 1M clusters and an average number of 8 DNA reads per cluster.

CPL

The CPL algorithm is a second reconstruction step used on clusters with low confidence from the DNN output. The algorithm receives a cluster of reads and its goal is to predict their encoded sequence. Since the clusters that are sent to the CPL algorithm have low confidence, often the DNN predictions of these clusters are not correct. Therefore, we built the CPL to not use either the inference created by the DNN nor the prior knowledge of the sequencing and synthesis error rates. The algorithm works directly on the clusters as they were obtained by the clustering algorithm.

First, the CPL algorithm takes the first read from the cluster, and then uses dynamic programming to calculate the edit distance of the first read from any of the other reads in the cluster. This step is used to estimate the errors that occurred in the reads of the cluster. Based on this calculation, the algorithm creates vectors of edit operations (deletions, insertions, and substitutions) that describe how to transform the first read to any of the other reads in the cluster using edit operations. These edit-operations vectors represent estimations of the number of occurrences of each edit error in the cluster and their location with respect to the first read. Next, the algorithm creates a directed acyclic graph based on these estimations of the errors. The vertices in the graph are the symbols of the first read, or the error events that were predicted in it based on the calculations. The edges in the graph connect any two vertices that correspond to symbols/errors which occur in adjacent locations. The weights of the edges are defined based on the occurrences of the two connected vertices in the edit-operations vectors. Finally, the algorithm returns the longest path, that represents the sequence with maximum probability based on the algorithm's estimations.

Confidence Filter and Safety Margin

One of the key features in our suggested solution is the confidence filter, which is a function that allows us to decide whether to rely on the DNN's output or not. The confidence filter is used for two purposes, the first is classifying highly erroneous DNN predictions as erasures, where in our pipeline, this equates to a missing cluster. This is beneficial as correcting erasures requires less redundancy than correcting substitutions of entire sequences. The second purpose is to decide which of the clusters should be sent to the CPL algorithm.

The main component of the confidence filter is a confidence function which operates on the soft output of the DNN. The soft output of the DNN is a $4 \times L$ matrix M, in which the 4 entries of the i-th column can be thought as the probabilities that the i-th symbol of the prediction is the corresponding symbol ACGT and L is the encoded sequence length.

In the confidence function we utilize this property, specifically, we use the arithmetic mean of the maximal value in each column, which represents the probability that the predicted symbol is correct. This mean value is defined as $m(M) \triangleq \frac{1}{L}\sum_{j=1}^{L} \max\{M_{1,j}, M_{2,j}, M_{3,j}, M_{4,j}\}$ where $M_{i,j}$ is the value of the i-th entry in the j-th column of M. Intuitively, smaller values of $m(M)$ correspond to cases in which the DNN is less confident in its output.

An additional property that is considered in the confidence filter is the cluster size. More precisely, our confidence filter considers the fact that the output of the confidence function of smaller clusters, i.e. lower SNR, tends to be lower in general and integrates this property to the

final filtering decision. The confidence function is defined as

$$c(M, \textit{cluster size}) \triangleq m(M)^{2 \cdot \textit{cluster size}}.$$

The confidence filter evaluates the following conditions:

1) If

$$c(M, \textit{cluster size}) \leq \textit{confidence}_{\textit{threshold}} \quad \text{and} \quad \textit{cluster size} \leq \textit{cluster size}_{\textit{threshold}}$$

the cluster is classified as an erasure (i.e. missing cluster).

2) Otherwise, if the CPL is incorporated in the pipeline,

$$c(M, \textit{cluster size}) \leq \textit{confidenc}_{\textit{threshold}} \text{ and } \textit{cluster size} > \textit{cluster size}_{\textit{threshold}}$$

then the cluster is passed to the CPL algorithm.

3) Otherwise, the confidence filter trusts the DNN output and passes it as it is to the decoder.

The optimization for the accuracy vs. runtime tradeoff is controlled by the values of $\textit{confidence}_{\textit{threshold}}$ and $\textit{cluster size}_{\textit{threshold}}$ and was performed using the pilot datasets. The performance of the DNN on the Illumina dataset was very good and did not require any other mechanism, therefor we set $\textit{confidence}_{\textit{threshold}} = 0$ and $\textit{cluster size}_{\textit{threshold}} = 3$. The performance of the DNN on the Nanopore single flowcell did require additional mechanisms to ensure successful information retrieval and robust safety margin. Hence, we chose a safety margin of 1%, which yielded the values of $\textit{confidence}_{\textit{threshold}} = 0.7$ and $\textit{cluster size}_{\textit{threshold}} = 4$. Evaluation was performed on the test datasets with the details provided in the Supplementary Information.

Supplementary Information for Deep DNA Storage: Scalable and Robust DNA-based Storage via Coding Theory and Deep Learning

Supplementary A: Results

A.1 Effects of the Data Structure on the Error Rate

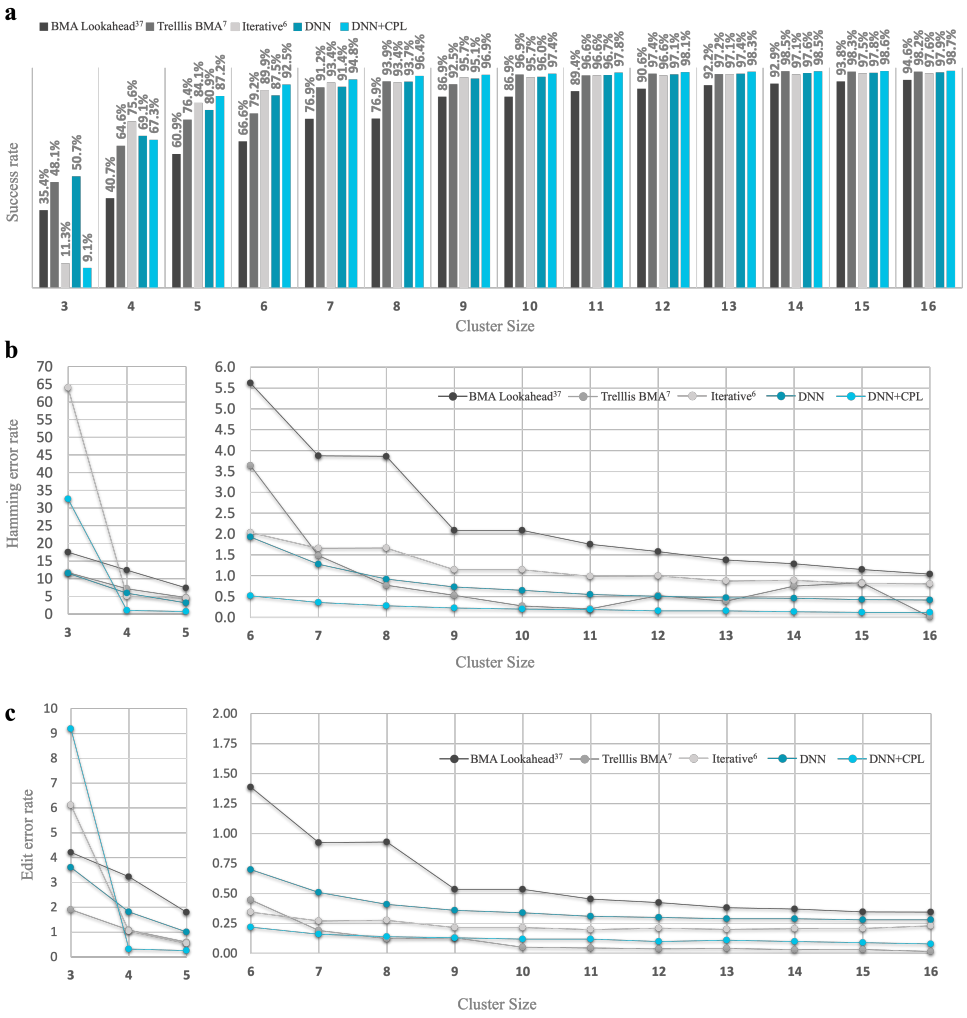
To test the robustness of our suggested solution, we compared the failure rates of each of the files separately. The results are summarized in Supplementary Table A.1 where we see that the DNAformer shows similar accuracy on both files. The failure rate measured in all three datasets is similar between the two files suggesting that the success of the retrieval pipeline of DNAformer does not depend on the entropy of the stored information.

File	Test dataset Illumina	Test dataset Nanopore single flowcell	Test dataset Nanopore two flowcells
File 1 - photo, audio, and text	0.056%	3.8%	1.61%
File 2 - random bits	0.056%	3.82%	1.49%

Supplementary Table A.1: Data modality effect on the failure rate. The results show our method is invariant to the data modality, implying that the DNAformer does not use the underlying structure in the data. Rather, we learn the noise characteristics of the channel and adapt to each type of synthesis-sequencing pair.

A.2 Effects of Cluster Size on the Error Rate

Evaluation of the reconstruction accuracy of the DNAformer as a function of the cluster size is provided in Supplementary Figure A.1. The results are shown on the test dataset, considering the read obtained by Nanopore MinION from two flowcells. For any cluster size $3 \leq t \leq 16$, we sample t copies from the obtained clusters that have at least t copies. The results of the reconstruction accuracy are shown by the failure rate, the Hamming error rate, and the edit error rate.



Supplementary Figure A.1: Cluster size effect on the error rate of the DNAformer. a, success rate as a function of the cluster size. We see the results improve as the cluster size increases, an attribute associated with the cluster's SNR. b, c, Hamming, and edit error rates as a function of the cluster size accordingly. The results show that larger clusters enhance the reconstruction accuracy, however, the enhancement decays when the cluster size reaches 16 and thus this size was selected as the maximal cluster size for the DNAformer. Furthermore, it can be seen that the accuracy is lower for clusters of size 4 and below, and thus size 4 was selected as the threshold for the confidence filter.

It can be seen that larger clusters correlate with higher accuracy in all of the tested algorithms. However, the improvement decays for clusters of size more than 15. For example, when comparing the success rate of clusters of size 15 with the success rate of clusters of size 16 the improvement of all the tested algorithms was less than 0.8%. Moreover, the improvement of the success rate of the DNAformer is less than 0.1%. Additionally, since the DNAformer was implemented on GPU, for inference efficiency purposes, the maximal cluster size was selected to be 16. It can be further seen that for clusters of size 4 and below, the results of most of the algorithms are relatively poor, up to 50% success rate, average edit distance of 1 to 9, and Hamming distance of at least 5. Thus, the cluster size threshold of the confidence filter was selected to be 4. Further details on the design of the confidence filter are provided in the appropriate subsection.

A.3 Analysis of the Clustering Step

Supplementary Table A.2 presents the results of the binning algorithm on our datasets. It can be seen that the binning algorithm clustered between 99%–100% of the reads that were obtained by Illumina sequencing, while only 26%–33% of the Nanopore reads were clustered successfully.

Dataset		Number of reads	Number of clustered reads	Number of clusters
Pilot	Illumina	528,636	528,636(100%)	1,000
	Nanopore	2,805,705	753,888(26.86%)	1,000
Test	Illumina	3,215,249	3,183,840(99%)	109,944
	Nanopore first flowcell	4,341,575	1,446,602(33.31%)	109,928
	Nanopore second flowcell	3,065,455	907,982(30%)	109,753
	Nanopore two flowcells	7,407,030	2,354,584(31.78%)	109,976

Supplementary Table A.2: Results of the binning algorithm on the different datasets. The right column is the number of non-empty clusters that were obtained by the algorithm out of $1,000$ for the pilot datasets and $110,000$ for the test datasets. In the second to right column, we see a high clustering percentage for the Illumina dataset while the Nanopore dataset exhibits lower clustering percentage due to its higher noise regime.

A.4 Analysis of the Accuracy Throughout the Retrieval Pipeline

Supplementary Table A.3 shows a detailed description of the reconstruction accuracy of each of the steps involved in the retrieval pipeline of the DNAformer as described in Figure 8.1. It can be seen that the total success rates of the DNAformer were between 96.12% and 99.94%. Furthermore, for the Nanopore reads, when using a single flowcell the number of wrong predictions was $5,550$ (5.05% of the clusters) and when using two flowcells the number of wrong predictions was $2,720$ (2.47% of the clusters). Our confidence filter removed 2.9% of the clusters (single flowcell) and 1.1% of the clusters (two flowcells), while the CPL was performed on 52.4% of the filtered clusters (single flowcell) and 91.75% of the filtered clusters (two flowcells).

Step	Test dataset Illumina	Test dataset Nanopore single flowcell	Test dataset Nanopore two flowcells
Number of tested clusters	109,944	109,976	109,928
Missing clusters	56	24	72
Wrong predictions of the DNN	6	2,720	5,550
Number of clusters filtered by the confidence filter	0	1,189	3,197
Number of clusters sent to the CPL algorithm	0	1,091	1,676
Number of clusters that were classified as erasures	0	98	1,521
Success rate from existing clusters	99.99%	98.34%	96.18%
Total success rate	99.94%	98.32%	96.12%

Supplementary Table A.3: Analysis of the accuracy throughout the retrieval pipeline. The table shows the reconstruction results in each of the different components of our reconstruction pipeline. The results show that for the Illumina test dataset and Nanopore test dataset two flowcells, the DNN is able to complete the reconstruction on its own. However, for the Nanopore test dataset single flowcell the DNN alone does not guarantee information retrieval. By using the confidence filter and CPL our method is able to cope with this SNR regime and guarantee successful retrieval.

A.5 Analysis of Coding Schemes

Supplementary Table A.4 shows a comparison of the coding schemes and design parameters that were used in previous DNA-based storage experiments. The error rates presented in the table are either based on our analysis if the datasets are publicly available, or on the reported error rates by the authors if the datasets are not publicly available. Note that since the calculation of the error rates depends on the exact clustering method that is used, it is possible to get a slightly different values for these rates. The results show that even though our channel error rates are an order of magnitude higher compared to the previous works that used Illumina sequencing, our work still achieves amongst the highest information rates.

The work by Yazdi et al. [16] was the first to use Nanopore sequencing for DNA-based storage. Although the technology was more error-prone than it is today, they were able to recover the stored information. We did not include this work in Figure 8.3 and Figure 8.4 as the length of their encoded sequences was considerably longer and the number of clusters was considerably smaller than other works, as seen in Supplementary Table A.4.

A.6 Comparison with the Trellis BMA Algorithm

Supplementary Table A.5 shows a comparison of our method to the Trellis BMA algorithm [7] and was performed separately due to its long running time. Therefore, we randomly selected 10,000 clusters from our Nanopore test datasets, each containing 16 reads. The running time of the trellis BMA algorithm on a 32 cores CPU was 61.2 hours, while the DNAformer runtime was 1.52 seconds on a single A40 GPU. It can be seen that the DNAformer outperforms the Trellis BMA algorithm in all of the tested parameters. More specifically, the DNAformer improves the failure rate by over 40%, while the edit error rate is improved by 3.5%, and the Hamming error rate is improved by over 50%.

Dataset	Data Size	Synthesis technology	Sequencing Technology	Encoded Sequences Length	Information rate (excluding primers)	Coding Technique	Channel error rate
Church et al. (2012) [10]	0.65MB	Agilent	Illumina HiSeq	159	0.83	Constrained Coding	0.74%
Goldman et al. (2013) [11]	0.63MB	Agilent	Illumina HiSeq	117	0.29	Constrained Coding	0.1774%
Grass et al. (2015) [12]	0.08MB	Custom Array	Illumina MiSeq	158	1.16	Constrained Codes + Inner-Outer Reed Solomon	1.06%*
Bronholt et al. (2016) [50]	0.15MB	Not reported	Illumina MiSeq	120	0.85	Hoffman code +XOR (outer)	$\sim 1\%$
Erlich and Zielinski (2017) [14]	2.11MB	Twist Bioscience	Illumina MiSeq	152	1.57	DNA fountain (Luby transform) + Reed Solomon	0.32%*
Yazdi et al. (2017) [16]	3KB	IDT	Nanopore MinION	$1,000$	1.74	Constrained codes + Deletion + Multiple sequence alignment)	$\sim 10\%$ –20%
Blawat et al. (2016) [51]	22MB	Agilent	Illumina MiSeq	230	1.08	Run length limited + Inner code CRC + Outer code RS	$< 0.5\%$
Organick et al. (2018) [15]	200MB	Twist Bioscience	Illumina NextSeq (Datasets) +Nanopore	150	1.1	Inner + Outer Reed Solomon	0.43%*
Chandak et al. (2020) [18]	11KB	Custom Array	Nanopore	165	0.063	Inner CRC/Convolutional code + Outer Reed Solomon Code	$\sim 10\%$
Wang et al. (2019) [17]	379.1MB	Twist Bioscience	Illumina HiSeq	190	1.67	CRC (single primer) + Repeat Accumulated Code	$\sim 0.02\%$
Anavy et al. (2019) [19]	6.4MB	Twist Bioscience (Composite)	Illumina MiSeq	194	1.96	Reed Solomon (inner) + Fountain Code + Composite	$< 0.1\%$
This Work	3.1MB	Twist Bioscience	Illumina MiSeq + Nanopore MinION	140	1.6/ 1.72 (no constraints)	Tensor Product Code + Flexible Block-based Constrained Code	4.47%*

Supplementary Table A.4: A comparison of coding methods, synthesis and sequencing technology, and design parameters of previous DNA storage experiments. We report that our work shows competitive results for the code rate while operating under higher channel error rate regimes than prior works. *The reported channel error rate is based on SOLQC [32].

Supplementary B: DNA Dataset

B.1 Publicly Available Datasets

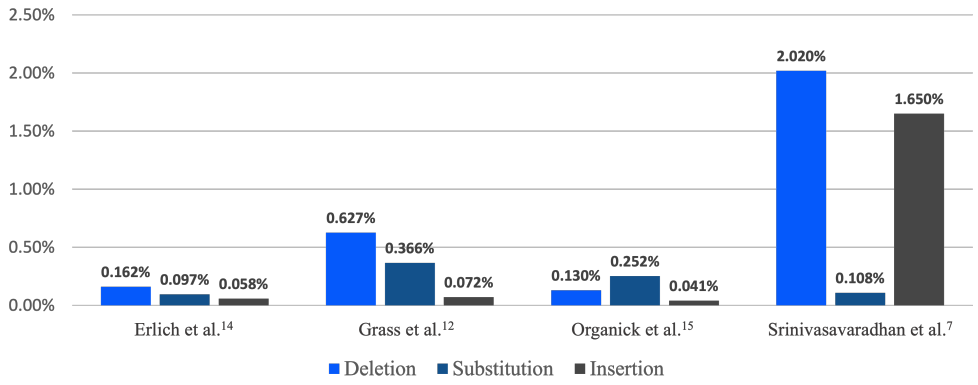
In this work, we compared the performance using several publicly available datasets. The number of encoded sequences in the dataset by Grass et al. [12] was $4,991$, while the pseudo-

	Trellis BMA	DNAformer
Failure rate	3.3%	1.97%
Edit error rate	0.2266%	0.219%
Hamming error rate	0.677%	0.313%

Supplementary Table A.5: Comparison of the reconstruction accuracy of Trellis BMA algorithm and the DNAformer. The table shows that the DNAformer improves the failure rate, the edit error rate, and the Hamming error rate of the Trellis BMA algorithm.

clustering algorithm obtained $4,982$ clusters. The total error rate of this dataset was 1.06%, and the average cluster size was 528. The dataset by Erlich et al. [14] includes $72,000$ clusters with a total error rate of 0.32% and an average cluster size of 178. The dataset by Organick et al. [15] contains $607,150$ encoded sequences, their reads were clustered into $596,244$ clusters with a total error rate of 0.43%. The average cluster size of this dataset was 32. Lastly, the dataset by Srinivasavaradhan et al. [7] consists of $9,954$ clusters (created from reads of $10,000$ encoded sequences) with a total error rate of 4.72% and an average cluster size of 17. It should be noted that in our reconstruction accuracy evaluations, the DNAformer used only up to 16 reads per cluster, while other algorithms used various larger cluster sizes. However, for runtime performance evaluations, we only compared clusters of size up to 16.

A detailed description of the error rate of the four different datasets can be found in Supplementary Figure B.2. The figure presents the substitution, insertion, and deletion rates of each of our tested datasets. It can be seen that in the data sets of Erlich et al. [14], Grass et al. [12], and Organick et al. [15], the most dominant errors were deletion and substitution, while in the data set by Srinivasavaradhan et al. [7], the most dominant errors were insertion and deletion.



Supplementary Figure B.2: Error rates of publicly available datasets. The figure shows the substitution, insertion and deletion rates of each of the tested data sets.

It should be also noted that there are three additional publicly available datasets that we do not include in our comparison. The work of Anavy et al. [19] presented a new approach for synthesis, in which the alphabet can be abstracted to include more than four symbols. This is done by utilizing composite letters that combine more than a single nucleotide in the same position, which makes their data less relevant for our pipeline. The work by Yazdi et al. [16]

consists of 17 encoded sequences, which is not enough for adequate error characterization by SOLQC [32], which is needed to train our DNN. Lastly, Chandak et al. [18] published a dataset that combines several different experiments together. As a result, their dataset contains multiple encoded sequences with the same index. This fact makes this dataset not suitable for our binning algorithm.

B.2 Our Illumina and Nanopore Datasets

Supplementary Figure B.3 shows the cluster size histogram for Illumina and Nanopore reads. The clusters were obtained using the binning step described in the Method section. All histograms are bell-shape. For the Illumina datasets, shown in Supplementary Figure B.3.a and Supplementary Figure B.3.b the bell-center of the histogram of the pilot is around 550, and around 40 for the test dataset. For the Nanopore datasets, the bell center of the pilot dataset, shown in Supplementary Figure B.3.c, is centered around 750. The Nanopore test datasets, for the case of a single flowcells and for the case of two flowcells, are shown in Supplementary Figure B.3.d and Supplementary Figure B.3.e and have the bell center around 12 and 21 accordingly.

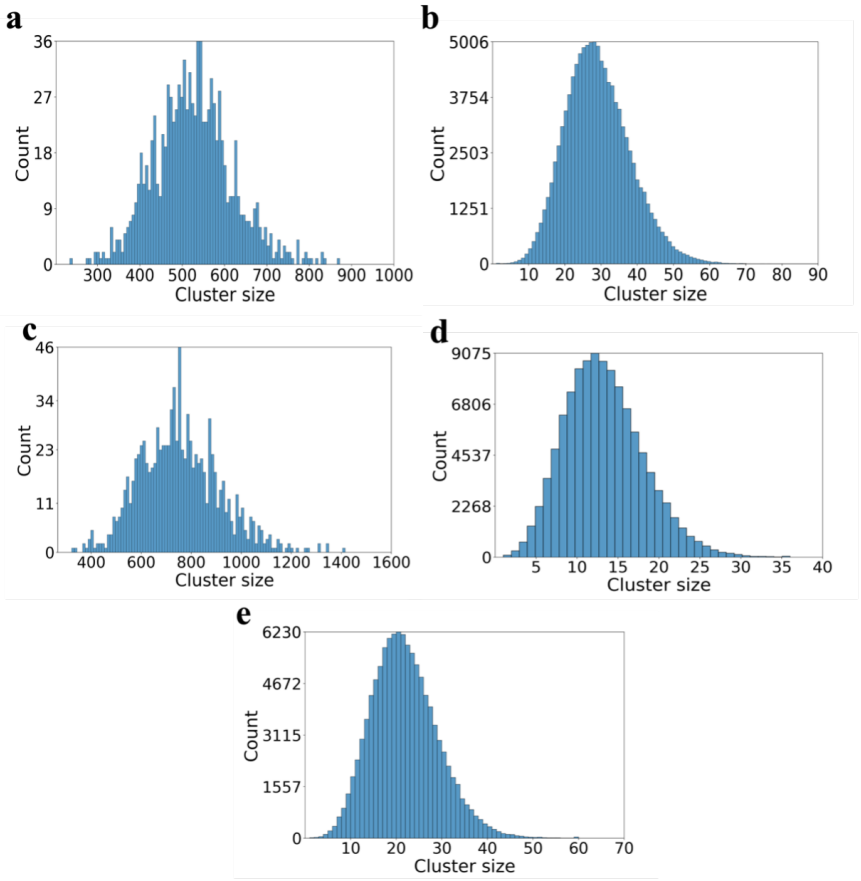
Supplementary C: Clustering

Following the sequencing process, the first step of the information retrieval process is clustering of the obtained reads. In this step, the obtained reads are partitioned into small groups known as clusters based on their origin. The process is known to be computationally difficult, and since the number of obtained reads is high, performing the clustering may require long running time, even when using efficient methods [21, 22].

The hash-based clustering method [22] takes a random short sequence of DNA, and partitions the reads based on the location of this randomly selected sequence. The clover clustering method [21] is based on a tree graph that is created from the reads. The edges in the tree estimate the distance between the reads and are used for clustering. Lastly, we also show the results of the perfect clustering method. In this method, we assume prior knowledge of the original encoded sequences, and each read is mapped into the cluster of its nearest encoded sequence. For our comparison, we used a brute force algorithm to create the clusters.

In our suggested method, we utilize the defined indices in each read to enable a faster and more efficient clustering process. Essentially, we perform simple binning based on each read's index, a fast and efficient process. The binning algorithm examines the 12 first bases (the index) of each read and bins it into clusters based on the inspected index. If the 12 first bases of a read do not match any of the possible indices, then the read is ignored. We compare our clustering method with previously published clustering algorithms and show that our retrieval pipeline is independent of our clustering method. That being said, using our binning method dramatically reduces the clustering time while preserving competitive reconstruction accuracy of the DNAformer.

An examination of different clustering methods is presented in Supplementary Table B.6. The evaluation was performed on a server with Intel(R) Xeon(R) CPU E5-2630 v3 2.40GHz.



Supplementary Figure B.3: Cluster size histograms of our data from Illumina and Nanopore datasets. For all plots, the X-axis shows the cluster size and Y-axis shows the number of clusters that were obtained at this size. a, Illumina pilot dataset. b, Illumina test dataset. c, Nanopore pilot dataset. d, Nanopore test dataset single flowcell. e, Nanopore test dataset two flowcells.

The results show the reconstruction failure rate and the runtime. It can be seen from the table that the running time of our binning algorithm is significantly smaller across all experiments.

It should be noted that the binning method considers the indices to obtain its clusters, while the perfect clustering method considers the encoded sequences to obtain its clusters. Therefore, when using these two methods, it is easy to match between an obtained cluster and an encoded sequence. However, when using the Hash-based clustering [22] and Clover clustering [21], a reconstruction algorithm should be applied on the clusters before matching them with valid indices. Although these two methods can create additional clusters without matching indices, our decoder is able to filter the nonvalid clusters.

Supplementary Table B.6 shows that for the Illumina test dataset, the failure rate of the DNAformer achieves almost the same results as the perfect clustering algorithm and the data can be retrieved by our decoder. Furthermore, in the Nanopore test dataset single flowcell, the

failure rate when using the binning method is higher compared to the perfect clustering and the Hash-based clustering [22], however, in all three cases, it is possible to retrieve the data. Finally, for the Nanopore dataset with two flowcells, the DNAformer failure rate is lower when using the binning algorithm compared to the perfect clustering. The Clover-based clustering method [21] does not allow for the successful retrieval of information on our datasets.

	Test dataset | Illumina			Test dataset | Nanopore single flowcell			Test dataset | Nanopore two flowcells		
Clustering method	Runtime (min)	No. of clusters	Failure rate	Runtime (min)	No. of clusters	Failure rate	Runtime (min)	No. of clusters	Failure rate
Binning	0.4	109,944	0.055%	0.36	109,928	4.79%	0.483	109,976	2.01%
Perfect	14,434	109,948	0.053%	12,232	109,970	3.8%	22,309	109,970	3.18%
Hash [22]	94	491,309	28.4%	124	218,681	3.05%	603	508,605	1.32%
Clover [21]	4	97,717	85.49%	8.54	299,190	44.34%	14.44	368,027	35.48%

Supplementary Table B.6: Comparison of different clustering methods. The results show that our binning method, although very simple and fast, achieves competitive failure rates with other methods. Showcasing how other components in the retrieval pipeline can operate with the additional noise. The results also show that although the Hash method creates more clusters than the original file contained, our decoder is able to filter the incorrect clusters and retrieve the information on the Nanopore datasets. For all clustering methods examined, the reconstruction was done with the DNAformer.

Supplementary D: Reconstruction

D.1 Model Architecture

The DNAformer architecture follows a Siamese structure with two branches and shared weights fused together to form a single unified predicted sequence, as shown in Figure 8.1.b. First an alignment module is applied whose purpose is to learn the required alignment for each read independently. The alignment module uses an Xception [48] inspired architecture with depth-wise separable 1D convolutions and multiple kernel heads with sizes of $1, 3, 5$, and 7. The purpose of using multiple kernels in the embedding layer is to allow the model to capture different shifts caused by deletion or insertion errors. The module is constructed with a repeatable block of a linear layer, followed by layer normalization and GELU activation. In addition, this module outputs a sequence with the required output length.

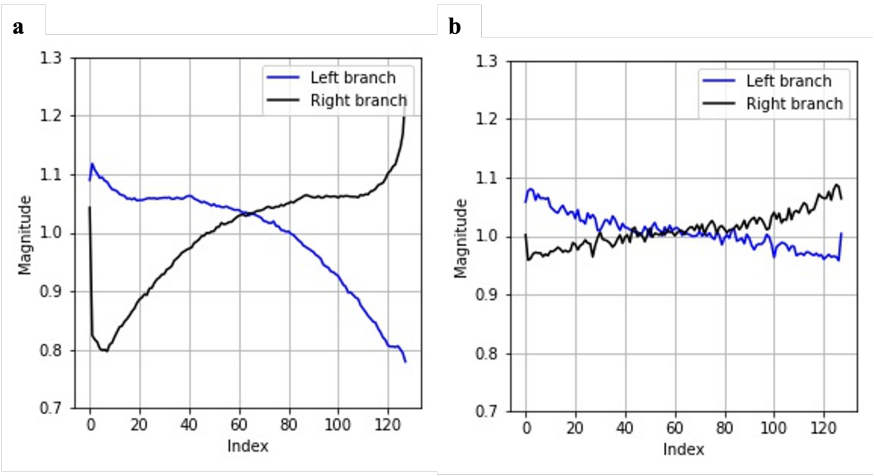
Following this module, we sum over the cluster dimension with the goal of improving the model's robustness to differences in cluster size. This operator has the effect of NCI and aims to increase the SNR of the data.

After the NCI aligner layer, the architecture employs an embedding module whose architecture is similar to the alignment module, however, now the operations are employed on the whole cluster and not on each read independently. The goal is to learn correlations between the different reads and prepare the data for the Transformer layer.

The transformer layer is a multi-head transformer architecture with linear layers for the feedforward part. We do not use position embeddings in this module. After the last transformer

block, a linear module is used to reduce the number of features to 4 which represents one-hot encoding for DNA representation.

The fusion vector has the length of the encoded sequence, and its purpose is to learn the optimal combination between the two branches. The fusion vector parameters are initialized with a constant value of 1 for each index, which means equal contribution from each branch. The results provided in Supplementary Figure D.4 show different behavior for each experiment reflecting that the model learns different optimal combinations for different sequencing technologies and noise patterns.



Supplementary Figure D.4: Fusion vector values. a Nanopore experiment. b Illumina experiment. The fusion vector learns to combine between the left and right branches in the DNN architecture and yields a single prediction sequence with the length of the required encoded sequence.

Supplementary Table D.7 shows the results of the ablation study conducted on the DNAformer architecture using the Nanopore test two flowcells dataset. The results show performance improvement as model capacity increases, coinciding with the results of DNNs for language and computer vision tasks. Additionally, we see that DNAformer trained on the pilot dataset statistics is able to achieve reconstruction of the full dataset. The Siamese architecture allows to introduce an additional loss mechanism to enforce consistency between the left and right branches and shows improvement other than using cross-entropy only.

Several data augmentation methods were examined. The first mechanism injects random false copies into the training process. This helps the model learn how to ignore such cases that occur due to clustering errors. The second mechanism varies the standard deviation of the generated noise statistics in the data generator during training. Lastly, the third mechanism controls each type of error (substitution, insertion, or deletion) separately to vary the generation process even further.

Architecture	Model capacity	Data generator	Loss	Augmentationsl					No. of errors
				Substitution deviation	Insertion deviation	Deletion deviation	STD deviation	False copies	
Single branch	20M	Nanopore test	CE	Off	Off	Off	Off	Off	8,456
Single branch	70M	Nanopore test	CE	Off	Off	Off	Off	Off	4,825
Single branch	70M	Nanopore test	CE	Off	Off	Off	0.1	Off	4,734
Single branch	70M	Nanopore test	CE	Off	Off	Off	Off	2	4,233
Single branch	70M	Nanopore test	CE	Off	Off	Off	0.1	2	3,754
Single branch	70M	Nanopore test	CE	75%	75%	75%	Off	Off	5,854
Single branch	70M	Nanopore test	CE	125%	125%	125%	0	Off	4,437
Single branch	100M	Nanopore pilot	CE	Off	Off	Off	0.1	2	3,362
Siamese	100M	Nanopore pilot	CE + Consistency	Off	Off	Off	0.1	2	2,896
Siamese	100M	Nanopore pilot	CE + Consistency	125%	125%	125%	0.1	2	2,720
Siamese + CPL	100M	Nanopore pilot	CE + Consistency	125%	125%	125%	0.1	2	1,842

Supplementary Table D.7: DNAformer ablation study. In this comparison, we focused on the Nanopore test two flowcells dataset. The examination included single branch and Siamese architecture where we see a clear benefit of using the latter. In addition, this architecture also allowed us to incorporate a consistency loss in parallel to the CE loss. From the results reported it is evident that the increase in model size also improves the results. Several data augmentation strategies were examined, and their collective contribution is reported. Lastly, combining the DNN with the CPL improves the results further, as additional small clusters are solved by the CPL.

D.2 Simulated Data Generation

In our comparison on publicly available datasets, we model the noise characteristics based on the dataset. Therefore, it is important to verify that there is no case of overfitting when reporting the results on these datasets. To verify this issue, we randomly split each dataset in half. One half of each dataset was used to generate the error statistics for our data generator and the other half was used for evaluation. In other words, the evaluation was performed on un-modelled data.

The results of this examination are presented in Supplementary Table D.8, where we used the single branch 70M parameters configuration for fast experimentation. We report a good fit between the evaluation on the full and split datasets, thereby supporting the results reported in Figure 8.3.

Assessment of the relationship between real and simulated data for training was performed using the dataset provided by Organick et al. [15] due to its relatively large size. The results are provided in Supplementary Table D.9 where we examined four data configurations. All configurations used the single branch 70M parameters configuration. To create the evaluation, the

	Erlich et al. [14]		Grass et al. [12]		Organick et al. [15]	
Synthesis	Twist Bioscience		CustomArray		Twist Bioscience	
Sequencing	Illumina miSeq		Illumina miSeq		Illumina NextSeq	
Dataset	Full	Split	Full	Split	Full	Split
Dataset size	72,000	36,001	4,989	2,495	596,499	298,249
Number of wrong predictions	16	7	64	31	1,373	630
Error percentage	0.02%	0.019%	1.3%	1.25%	0.23%	0.21%

Supplementary Table D.8: Performance comparison between full and split datasets. The results show a good fit between the full and split dataset cases. The split dataset case used half of the data to generate error statistics for the data generator and evaluated on the other half of unmodelled data. Wrong prediction is defined as having at least one wrong character out of the entire predicted sequence.

dataset was split into 425,006 frames for validation and 141,668 frames were used to model the error statistics and were also used to train the 'Real data' configuration. The 'label + simulated data' configuration used real DNA designed strands as labels and generated simulated reads at each iteration. The 'mixed real and simulated data' configuration utilized a linear, progressive blending between the two data sources, which started with simulated data only in the first epoch and ended with real data only in the last epoch. The results show that utilizing our data generator cannot only replace real data, which is expensive and time-consuming to acquire but also improve performance, largely due to the unlimited amount of simulated data that can be generated. Furthermore, the results show a small improvement when combining a small amount of real data and a large amount (x10) of simulated data.

	Simulated data	Labels + simulated data	Real data	Mixed real and simulated data
Number of wrong predictions	988	1,155	1,460	948
Failure rate	0.23%	0.27%	0.34%	0.22%

Supplementary Table D.9: Comparison between real and synthetic data performance. The results show the proposed data generator achieves better performance than real data-based training. The combination of real and simulated data achieved the best performance. Wrong prediction is defined as having at least one wrong character out of the entire predicted sequence.

D.3 The CPL Algorithm

In this subsection, we first give a high-level description of the computational steps of the CPL algorithm:

1) **Input.** The algorithm receives a cluster, denoted by $\mathbf{C}$ of t reads, and the encoded sequence's length u and estimates the original encoded sequence of $\mathbf{C}$.

2) **Edit distance calculation.** The algorithm considers the first read of $\mathbf{C}$, denoted by $\boldsymbol{y}_1$

and uses dynamic programming to calculate the selected reads' edit distance from any other read in the cluster. This step involves the calculation of the edit distance of $t-1$ pairs of reads.

3) **Edit distance calculation.** The algorithm uses a backtracking technique on the edit distance calculation to retrieve edit-operation vectors that describe how to change $\boldsymbol{y}_1$ to any of the other reads in the cluster.

4) **Edit distance graph generation.** The algorithm utilizes the set of edit-operation vectors to form a graph, in which the vertices set consists of two types of vertices; The first type corresponds to the symbols and their locations as they appear in the selected read, while the second type is correlated with possible insertion operations between the symbols of the selected read. The edges connect between consecutive (by index) vertices of the first type and adjacent vertices of the two types related to an insertion operation. The weights of the edges are defined as the logarithm of the estimated probability of observing the edge and its two consecutive vertices as part of a path that corresponds to the algorithm's output.

5) **Longest path calculation.** Finally, the longest path in the graph is computed, which corresponds to a sequence of length u, and returns its corresponding sequence as the algorithm's output estimation.

A detailed description of the CPL algorithm is provided below.

Input and Edit Distance Calculations.

The algorithm chooses the first reads in the cluster and denotes it by $\boldsymbol{y}_1$. Using a dynamic programming technique [52], we calculate the edit distance between $\boldsymbol{y}_1$ and each of the other reads in the cluster $\boldsymbol{y}_k$ for $2 \leq k \leq t$. To compute the edit distance, the algorithm creates a $|\boldsymbol{y}_1| \times |\boldsymbol{y}_k|$ dynamic programming matrix, in which the (i, j)-th cell of the matrix represents the edit distance between the first i symbols in $\boldsymbol{y}_1$ and the first j symbols in $\boldsymbol{y}_k$. Next, the algorithm uses a backtracking method on the computed matrices to retrieve for each pair $\boldsymbol{y}_1$ and $\boldsymbol{y}_k$ a vector of edit operations that describes how to obtain $\boldsymbol{y}_k$ from $\boldsymbol{y}_1$. This vector is denoted by $\mathbf{EV}(\boldsymbol{y}_1, \boldsymbol{y}_k)$, and the set of all such vectors is denoted by $\mathbf{EV}_{\boldsymbol{y}_1}(\mathbf{C})$.

Edit-Operations Vectors Preparation.

There are four possible operations: copy, insertion, deletion, and substitution. Note that insertion operations can cause various lengths for the vectors in $\mathbf{EV}_{\boldsymbol{y}_1}(\mathbf{C})$, and thus to avoid this confusion, we partition each edit-operation vector into two vectors of length $|\boldsymbol{y}_1|$ and $|\boldsymbol{y}_1|+1$. The first vector is denoted by $\mathbf{EV}_{CDS}(\boldsymbol{y}_1, \boldsymbol{y}_k)$, and it corresponds to copy, substitution, and deletions operations. That is, for $1 \leq i \leq |\boldsymbol{y}_1|$, the i-th entry of the vector $\mathbf{EV}_{CDS}(\boldsymbol{y}_1, \boldsymbol{y}_k)$ describes whether the i-th symbol of $\boldsymbol{y}_1$ should be copied, deleted, or substituted with another symbol in order to transform $\boldsymbol{y}_1$ to $\boldsymbol{y}_k$. The second vector is denoted by $\mathbf{EV}_I(\boldsymbol{y}_1, \boldsymbol{y}_k)$ and it describes the required insertion operations that need to be performed on $\boldsymbol{y}_1$ to transform it to $\boldsymbol{y}_k$. That is, for $1 \leq i \leq |\boldsymbol{y}_1|$, the i-th entry of $\mathbf{EV}_I(\boldsymbol{y}_1, \boldsymbol{y}_k)$ describes the symbol(s) that should be

inserted to $\boldsymbol{y}_1$ before its i-th symbol to transform it to $\boldsymbol{y}_k$. For simplicity, the possible symbols also include the empty word ε, which corresponds to the case where no insertion is required. It should be mentioned that for given $\boldsymbol{y}_1$ and $\boldsymbol{y}_k$, the vector $\mathbf{EV}(\boldsymbol{y}_1, \boldsymbol{y}_k)$ is not necessarily unique, and in this case the algorithm chooses one of them randomly.

Example of Edit-Distance Calculation and Edit-Operations Vectors.

Supplementary Figure D.5 provides an example that describes the dynamic programming matrix for two possible reads $\boldsymbol{y}_1 = \mathsf{ATTACA}$ and $\boldsymbol{y}_k = \mathsf{ATGCTG}$.

-	-	A	T	T	A	C	A
-	0	1	2	3	4	5	6
A	1	0	1	2	3	4	5
T	2	1	0	1	2	3	4
G	3	2	1	1	2	3	4
C	4	3	2	2	2	2	3
T	5	4	3	2	3	3	3
G	6	5	4	4	3	3	3

Supplementary Figure D.5: Example of edit-distance calculation and edit-operations vectors. In this example of a dynamic programming matrix, $\boldsymbol{y}_1 = \mathsf{ATTACA}$ and $\boldsymbol{y}_k = \mathsf{ATGCTG}$.

By backtracking the matrix described in Supplementary Figure D.5, we can derive the following two error vectors as defined above:

$$\mathbf{EV}_{CDS}(\boldsymbol{y}_1, \boldsymbol{y}_k) = ((0, copy, \mathsf{A}), (1, copy, \mathsf{T}), (2, del, \mathsf{T}), (3, sub, \mathsf{A} \to \mathsf{G}), (4, copy, \mathsf{C}), (5, copy, \mathsf{A}))$$
$$\mathbf{EV}_{I}(\boldsymbol{y}_1, \boldsymbol{y}_k) = ((0, ins, \varepsilon), (1, ins, \varepsilon), (2, ins, \varepsilon), (3, ins, \varepsilon), (4, ins, \varepsilon), (5, ins, \varepsilon), (6, ins, \mathsf{G}))$$

Estimations of the Error Probabilities.

Next, the algorithm uses the set of error vectors $\mathbf{EV}_{\boldsymbol{y}_1}(\mathbf{C})$ to estimate the probability of having a specific symbol in a specific index in the algorithm's estimation. For each index $1 \leq i \leq n$ and for any two symbols $\sigma, \sigma' \in \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\} \cup \{\emptyset\} \cup \{\$\}$ we define the conditional probability to obtain symbol σ in index $i+1$, given that the i-th symbol is σ'. The symbol in the 0-th index is strictly defined as $\$ \notin \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$ to mark that this is the beginning of the word, and the symbol $\emptyset$ corresponds to the empty word. The estimated conditional probability is given in the expression:

$$p_i^{\boldsymbol{y}_{j'}\mathbf{C}}(\sigma|\sigma') \triangleq \frac{\left|\left\{\boldsymbol{v} \in \mathbf{EV}_{\boldsymbol{y}_j}(\mathbf{C}) \ : \ \boldsymbol{v}[i+1] = \sigma \text{ and } \boldsymbol{v}[i] = \sigma'\right\}\right|}{\left|\left\{\boldsymbol{v} \in \mathbf{EV}_{\boldsymbol{y}_j}(\mathbf{C}) \ : \ \boldsymbol{v}[i] = \sigma'\right\}\right|}$$

where $\boldsymbol{v}[i]$ refers to the i-th symbol of the vector $\boldsymbol{v}$. Following that, we calculate the set:

$$Post_{i,\sigma'}^{\boldsymbol{y}_{j'}\mathbf{C}} \triangleq \left\{\sigma \ : \ \sigma \in \mathbf{EV}_{\boldsymbol{y}_j}(\mathbf{C})[i+1], P_i^{\boldsymbol{y}_{j'}\mathbf{C}}(\sigma|\sigma') \neq 0\right\},$$

which is the set of the symbols that can be achieved on the $(i+1)$-th index of an error vector in $\mathbf{EV}_{\boldsymbol{y}_j}(\mathbf{C})$, given that the i-th symbol was σ'. Note that, the set $Post_{i,w\sigma'}^{\boldsymbol{y}_j,\mathbf{C}}$ can be empty.

Graph Generation.

Based on these estimated probabilities, we can now define the edit graph $G_{\boldsymbol{y}_j,\mathbf{C}} = \Big(V_{\boldsymbol{y}_j}, E_{\boldsymbol{y}_j}\Big)$ of a read $\boldsymbol{y}_j$ and a cluster $\mathbf{C}$. The edit graph of the sequence is defined as follows.

1) $V_{\boldsymbol{y}_j} = \Big\{\sigma' \,:\, \sigma' \in \mathbf{EV}_{\boldsymbol{y}_j}(\mathbf{C})[i], 1 \leq i \leq 2|\boldsymbol{y}_j|+1, \sigma' \neq \emptyset\Big\} \cup \{\$\}$. The vertices are defined as the (non-empty word) symbols from the sets $\mathbf{EV}_{\boldsymbol{y}_j}(\mathbf{C})$. Note that, the same symbol can appear as more than one vertex, depending on its indices in the vectors of $\mathbf{EV}_{\boldsymbol{y}_j}(\mathbf{C})$. Furthermore, in odd indices of vectors in $\mathbf{EV}_{\boldsymbol{y}_j}(\mathbf{C})$, there can be more than one symbol. In this case, we have a vertex for each symbol that appears in this position of the vector, where the symbols are connected to each other based on their order, where the weight of their edges is 0.

2) $E(\boldsymbol{y}_j) = \Big\{(\sigma',\sigma) \,:\, \sigma' \in \mathbf{EV}_{\boldsymbol{y}_j}(\mathbf{C})[i], \sigma \in Post_{i,w}^{\boldsymbol{y}_j,\mathbf{C}}, \sigma \neq \emptyset\Big\}$. Each edge connects between any two vertices that represent symbols that appear consecutively in an error vector in the set $\mathbf{EV}_{\boldsymbol{y}_j}(\mathbf{C})$. In case σ' or σ consists of more than one symbol, we connect the last symbol of σ' to the first symbol of σ.

3) $W : E_{\boldsymbol{y}_j} \to \mathbb{R}$ is a weight function, defined on the edges. The weight of an edge $(\sigma',\sigma)_i \in E_{\boldsymbol{y}_j}$ is defined as the log of its conditional probability, that is

$$W((\sigma',\sigma)_i) = \log\Big(P_i^{\boldsymbol{y}_j,\mathbf{C}}(\sigma|\sigma')\Big).$$

4) For $\sigma' \in \mathbf{EV}_{\boldsymbol{y}_j}(\mathbf{C})[i]$ where $\emptyset \notin Post_{i,\sigma'}^{\boldsymbol{y}_j,\mathbf{C}}$, we connect the last symbol of σ' to the first symbol of any $\sigma \in Post_{i+1,\emptyset}^{\boldsymbol{y}_j,\mathbf{C}}$, $\sigma \neq \emptyset$. The weight of this edge is $\log(P_i(\emptyset|\sigma')) + \log(P_i(\sigma|\emptyset))$. The same holds for any index $j \geq i$, in this case, the last symbol of σ' is connected to the first nonempty symbol of any $\sigma \in Post_{j,\emptyset}^{\boldsymbol{y}_j,\mathbf{C}}, \sigma \neq \emptyset$. In this case, the weight of the edge is defined as

$$\log\big(P_i(\emptyset|\sigma')\big) + \log(P_{i+1}(\emptyset|\emptyset)) + \ldots + \log\big(P_{j-1}(\emptyset|\emptyset)\big) + \log\big(P_j(\sigma|\emptyset)\big).$$

5) The vertex represents the beginning of the sequence, and is connected to any vertex σ that corresponds to index $i = 1$ in the error vectors in $\mathbf{EV}_{\boldsymbol{y}_j}(\mathbf{C})$, with an edge of weight exactly $\log(P_1(\sigma|\$))$.

Output of the Algorithm.

Finally, the CPL algorithm calculates the heaviest path in the edit graph $G_{\boldsymbol{y}_j,\mathbf{C}} = \Big(V_{\boldsymbol{y}_j}, E_{\boldsymbol{y}_j}\Big)$ of length $u+1$. If there does not exist a path of length $u+1$, the algorithm picks the one that

is closest in length to $u+1$ edges. If there is more than one, it picks one of them randomly. We denote the number of edges in the selected path as $\hat{u}$. The $\hat{u}$ vertices in this path correspond to $\hat{u}-1$ symbols and one vertex of the symbol \$. Therefore, the path induces a sequence $\hat{\boldsymbol{x}}$ that estimates the original sequence of the cluster. The output of the algorithm is $\hat{\boldsymbol{x}}$. The complexity of the algorithm is $\mathcal{O}(tu^2)$ when t is the number of reads and u is the encoded sequence's length.

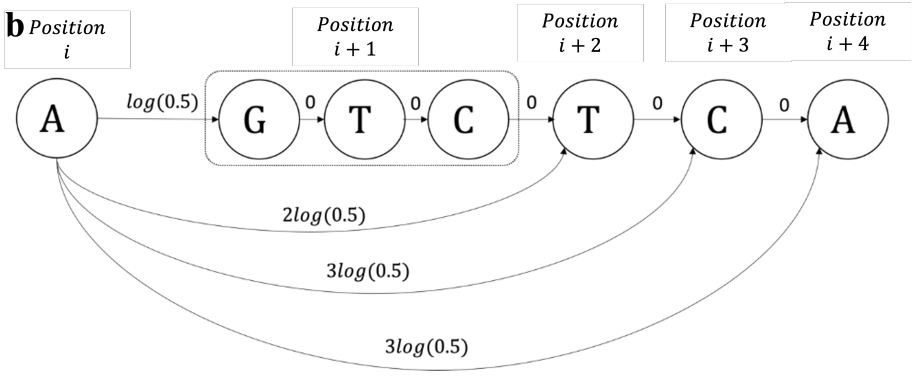
Example of the Edit Graph.

Supplementary Figure D.6 depicts an example of the edit graph that is used by the CPL algorithm. In this example, we describe a possible input to the CPL algorithm and its corresponding graph. We assume a read which is denoted by $\boldsymbol{y}_j \in \mathbf{C}$ is given to the CPL algorithm. The example describes only five positions in $\boldsymbol{y}_j$, whose indices are given by $i, i+1, i+2, i+3, i+4$. For each of these positions, the specific symbol of $\boldsymbol{y}_j$ is given, together with the observed symbols in the set $Post_{i,\sigma'}^{\boldsymbol{y}_{j'},\mathbf{C}}$. The set $Post_{i,\sigma'}^{\boldsymbol{y}_{j'},\mathbf{C}}$, in each of the described positions is calculated by the CPL algorithm. Finally, the graph which corresponds to the described input and to the sets $Post_{i,\sigma'}^{\boldsymbol{y}_{j'},\mathbf{C}}$ is depicted.

a

Position i		*Position i + 1*		*Position i + 2*		*Position i + 3*		*Position i + 4*	
String	*Post*	*String*	*Post*	*String*	*Post*	*String*	*Post*	*String*	*Post*
A	$\{GTC, \emptyset\}$	*GTC*	*T*	*T*	*C*	*C*	*A*	*A*	$\emptyset$
		$\emptyset$	$\{T, \emptyset\}$	$\emptyset$	$\{C, \emptyset\}$	$\emptyset$	*A*		

b



Supplementary Figure D.6: Example of the graph of the CPL algorithm. a, example of input read $\boldsymbol{y}_j \in \mathbf{C}$ of the CPL algorithm. Each column corresponds to a specific position of the input $\boldsymbol{y}_j \in \mathbf{C}$, where the positions are indexed by $i, i+1, i+2, i+3, i+4$. For each of these positions, the first column presents the observed string, denoted by *String* i, and the second column presents the values of the set $Post_{i,\sigma'}^{\boldsymbol{y}_{j'},\mathbf{C}}$ for $\sigma' \in$ *String* i, which is the set of possible sequences that are adjacent (from right) to the symbols that appear in the i-th position. The set $Post_{i,\sigma'}^{\boldsymbol{y}_{j'},\mathbf{C}}$ is calculated as defined in the CPL algorithm. b, shows the edit graph based on the table in a.

Supplementary E: Coding Scheme

Previous works in DNA-based storage have primarily relied on either inner codes, outer codes, or a combination of both. Inner codes involve encoding each sequence separately using an ECC, which can overcome up to d errors at the nucleotide level. Outer codes, on the other hand, involve encoding the sequences collectively to overcome missing or corrupted encoded sequences by adding redundant ones. However, adding redundancy for each encoded sequence can result in suboptimal use of redundancy, especially when d is chosen to address the worst-case scenario of reads. Conversely, if d is too small, many encoded sequences may be decoded incorrectly, requiring a high level of redundancy in the outer code to correct them. To overcome this tradeoff and utilize the inherent redundancy more effectively, we propose using a TP based coding scheme that can correct up to d errors in up to a fraction p of the encoded sequences. This allows a significant reduction in the number of redundancy symbols used for error correction while maintaining error-free retrieval of information.

Our suggested coding scheme encodes a block of b bits into $M = 55,000$ encoded sequences of length $L = 140$. The parameter b depends on the specific setup in which our constrained code is applied. Here we propose two different setups, the first enforces the length of the longest homopolymer to be at most 4 and the GC content of each encoded sequence to be between 30% to 70%. The second setup does not enforce any constraint on the encoded sequences. For the first setup, the total number of information bits is $b = 12,317,012$ bits (1.53963MB), and for the second setup $b = 13,249,024$ bits (1.656128MB).

Each strand is composed of 12 bases for index, including a single base that serves as a file identifier and $u = 128$ bases for data and redundancy as presented in Supplementary Figure E.7.f. A high-level description of our coding scheme is given in Supplementary Figure E.7.a-e. Our coding scheme is based on the following components:

1) **Index encoding.** A mapping function $E_{Ind} : [M] \to \{\mathsf{A},\mathsf{C},\mathsf{G},\mathsf{T}\}^{11}$ which is used to create the indices for the encoded sequences, i.e., for any $i \in \{0,\ldots,M-1\}$, the index of the i-th encoded sequence is $E_{Ind}(i)$. Here we designed the mapping E_{Ind} such that the indices of the different encoded sequences are far enough from each other and satisfy the constraint that the length of every homopolymer is at most 4. The index itself is of length $L-u = 12$. The additional base, which is the file identifier, is added later to satisfy some structural constraints. More details can be found under the Index encoding subsection.

2) **Diagonal-column encoding.** A mapping $E_{DRS}:\{0,1\}^{(M-r_1)\times(16\cdot 13)} \to \{0,1\}^{M\times(16\cdot 13)}$. The mapping E_{DRS} encodes the columns of a binary $(M-r_1)\times 16$ matrix diagonally to protect from any remaining errors that were not corrected using the information retrieval process or the other coding components. This code serves as a fail-safe mechanism, and by designing other components carefully, its redundancy, r_1, can be relatively low. For more details see the Diagonal-column encoding subsection.

3) **Constrained code.** A mapping function $E_{Cons} : \{0,1\}^{n_1} \to \{\mathsf{A},\mathsf{C},\mathsf{G},\mathsf{T}\}^{n_2}$ such that $\frac{n_1}{2} \le n_2 \le n_1$. The mapping E_{Cons} encodes a binary sequence of length n_1 into a length n_2 sequence over $\{\mathsf{A},\mathsf{C},\mathsf{G},\mathsf{T}\}$. In this work, we suggest a possible mapping that

satisfies two constraints: (1) GC-content between 45% and 55% with high probability and between 30% and 70% in the worst case, and (2) homopolymers of length 4 or less. For more details see the Constrained code subsection.

4) **Tensor-product code.** Our method utilizes a TP code, which involves a RS code with redundancy r_3 and a Bose–Chaudhuri–Hocquenghem (BCH) code [13] with redundancy r_2. Our TP code can be described as a mapping

$$E_{TP} : \mathcal{M}(M, u, r_2, r_3) \to \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}^{M \times u}$$

such that for $r_2 \leq M$, and $r_3 \leq u$, $\mathcal{M}(M, u, r_2, r_3)$ is the set of all $M \times u$ matrices over $\{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$ in which the bottom-right $r_3 \times r_2$ sub-matrix is empty. For more details about the applicable parameters, encoding, and decoding algorithms for E_{TP} see Tensor-product code and Decoding subsections.

E.1 Encoding Description

Our encoding scheme encodes a binary data $\boldsymbol{x}$ into a codeword matrix $X \in \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}^{M \times L}$. The redundancy parameters that were used in our dataset are $r_1 = 3,026, r_2 = 16$, and $r_3 = 4,786$. The number of bits in $\boldsymbol{x}$ is denoted by b, where $b = 238 \cdot (M - r_3) + 208 \cdot (r_3 - r_1)$ in our constrained setup, and $b = 256 \cdot (M - r_3) + 224 \cdot (r_3 - r_1)$ in our non-constrained setup. In the description to follow we use the notation $b = b_\ell \cdot (M - r_3) + b_s \cdot (r_3 - r_1)$ where b_ℓ and b_s should be set according to the desired constrained setup. The encoding of $\boldsymbol{x} \in \{0,1\}^b$ into $X \in \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}^{M \times L}$ is done by the following steps.

1) **Index encoding.**

   (a) Define an $M \times (L - u)$ matrix $\mathcal{I}$, where for any $i \in [M]$, fill the i-th row of $\mathcal{I}$ with the index $E_{Ind}(i)$.

   (b) Create an empty $M \times b_\ell$ matrix $\mathcal{X}_b$ and concatenate it to the right of the matrix $\mathcal{I}$. The output of this step is depicted in Supplementary Figure E.7.a.

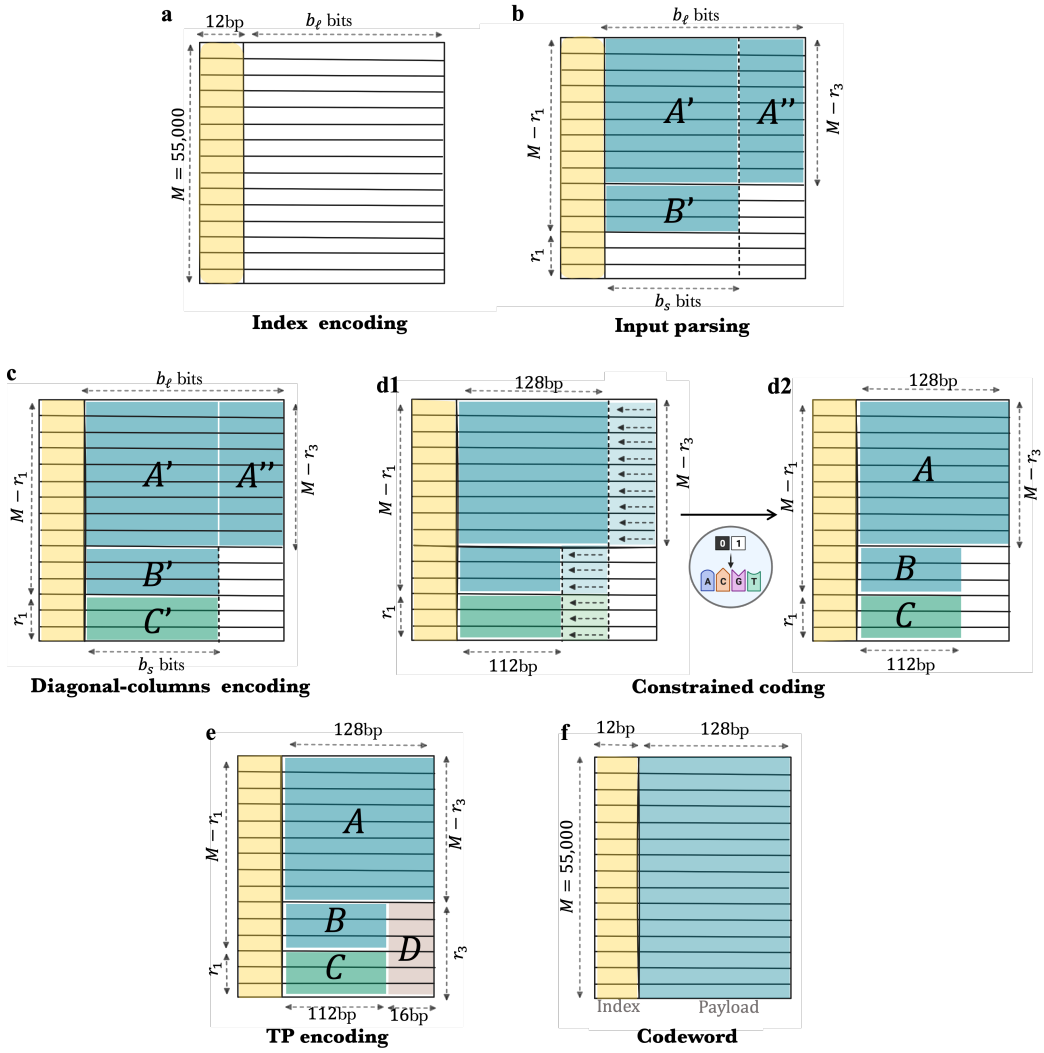
2) **Index parsing.**

   (a) Fill the matrix $\mathcal{X}_b$ with the bits of $\boldsymbol{x}$, such that the first $M - r_3$ rows contain exactly b_ℓ bits each, and the next $r_3 - r_1$ rows contain b_s bits each (the rest of the matrix remains empty).

   (b) Denote by A', A'' and B' the following submatrices of $\mathcal{X}_b$.

   - A' is the top-left $(M - r_3) \times b_s$ submatrix of $\mathcal{X}_b$.
   - A'' is the top-right $(M - r_3) \times (b_\ell - b_s)$ submatrix of $\mathcal{X}_b$.
   - B' is the $(r_3 - r_1) \times b_s$ submatrix of $\mathcal{X}_b$ located just below A'.

   This partition of $\mathcal{X}_b$ is depicted incSupplementary Figure E.7.b.



Supplementary Figure E.7: Encoding scheme. The main steps in our encoding scheme are shown. a, index encoding. Assigning each row, a unique identifier. b, input parsing. Organizing the binary data into a specific structure within a matrix form. c, diagonal-columns encoding. Using an RS code in a diagonal manner to encode A' and B'. d1-2, constrained coding. Convert the binary data into base 4 (corresponding to $\{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$ molecules) while satisfying design constraints. e, TP coding. The main component that is responsible for error-correcting. Adds additional redundancy symbols to protect A, B, and C matrices against substitution errors. f, codeword. In the output of our coding scheme, each row serves as an encoded sequence.

3) **Diagonal-columns encoding.**
Encode the $(M - r_1) \times b_s$ submatrix that is composed of A' and B' using E_{DRS}. This step adds r_1 redundancy bits to each column and the submatrix that corresponds to these redundancy bits is denoted by C' as depicted in Supplementary Figure E.7.c.

4) **Constrained code.**
Encode each of the rows of $\mathcal{X}_b$ using E_{Cons}. To this end, we use two instances of E_{Cons} as follows.

   (a) Encode the first $M - r_3$ rows (longer rows) we use E_{Cons} with parameters $n_1 = b_\ell$ and $n_2 = 128$.

   (b) Encode the last r_3 rows (shorter rows), we use E_{Cons} with parameters $n_1 = b_s$ and $n_2 = 128 - r_2$.
   The output of this step is presented in Supplementary Figure E.7.d1, where the 4-ary representation of A',A'' is denoted by A, the 4-ary representation of B' is denoted by B, and the 4-ary representation of C' is denoted by C.

   (c) Remove the $b_\ell - 128$ columns of the matrix $\mathcal{X}_b$ to obtain the $M \times u$ matrix $\mathcal{X}$. Note that $\mathcal{X}$ is a matrix over $\{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$ of size $M \times u$ in which the right-bottom $r_3 \times r_2$ submatrix is empty; see Supplementary Figure E.7.d2.

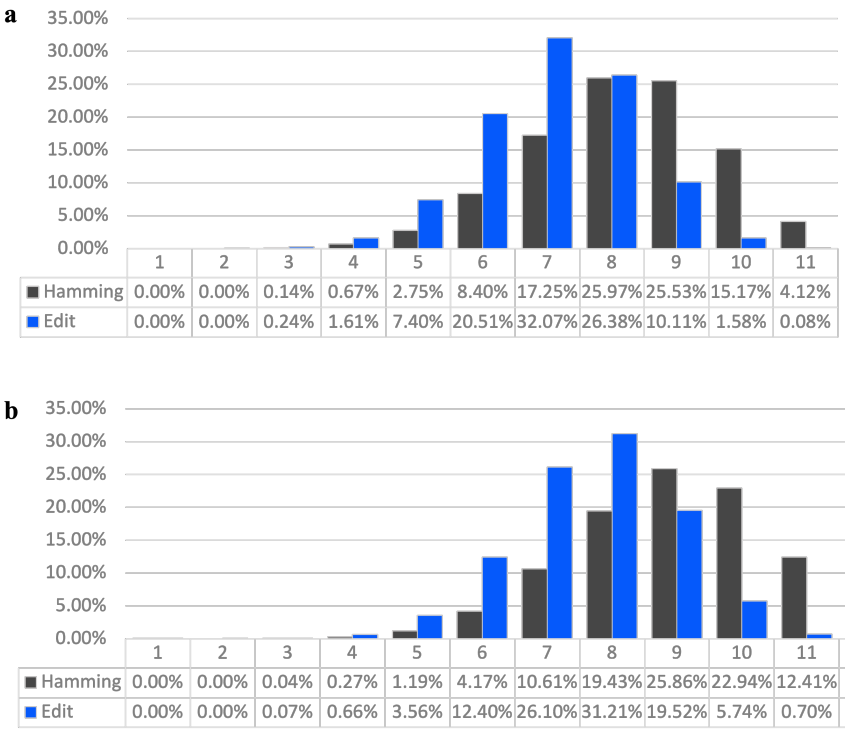
5) **TP encoding.** Encode the matrix $\mathcal{X}$ using E_{TP} to complete the $r_3 \times r_2$ missing symbols as depicted in Supplementary Figure E.7.e.

E.2 Index Encoding

The set of $M = 55,000$ indices of length 11 were selected randomly from the set of all sequences of length 11 over $\Sigma = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$, such that the maximal homopolymer's length is 4. To further demonstrate the robustness of our method, we used the same set of $M = 55,000$ indices for both files. To distinguish between the two files, we extended the index with an additional base, that serves as a file identifier (which is the 12-th symbol of the index). The file identifier for File 1 is either A or C, while the file identifier for File 2 is either G or T. By using the set of indices for the two files, we deliberately created a small number of pairs of indices with Hamming and edit distance equal to one. The results presented in this work show that our information retrieval pipeline allows for a small number of close indices which makes the design of the indices a much simpler task.

The exact file identifier is selected such that the homopolymer constraint is maintained (for more details see the Constrained code subsection). The set of $M = 55,000$ indices of length 11 and the set of $110,000$ indices of length 12 that were used for File 1 and File 2 can be found in the code repository published with this work.

Supplementary Figure E.8 presents a histogram of the edit and Hamming distances between pairs of indices in our set of indices. The results for the $M = 55,000$ indices of length 11 are given in Supplementary Figure E.8.a and the results for the $110,000$ indices that were used in our dataset are given in Supplementary Figure E.8.b. Even though we selected our indices randomly, most of the pairs are at Hamming and edit distance at least 5 from each other which is sufficient for our binning algorithm. This behavior was obtained by selecting the length of the indices to be sufficiently large with respect to the number of indices our scheme requires.



a	1	2	3	4	5	6	7	8	9	10	11
Hamming	0.00%	0.00%	0.14%	0.67%	2.75%	8.40%	17.25%	25.97%	25.53%	15.17%	4.12%
Edit	0.00%	0.00%	0.24%	1.61%	7.40%	20.51%	32.07%	26.38%	10.11%	1.58%	0.08%

b	1	2	3	4	5	6	7	8	9	10	11	12
Hamming	0.00%	0.00%	0.04%	0.27%	1.19%	4.17%	10.61%	19.43%	25.86%	22.94%	12.41%	3.09%
Edit	0.00%	0.00%	0.07%	0.66%	3.56%	12.40%	26.10%	31.21%	19.52%	5.74%	0.70%	0.03%

Supplementary Figure E.8: Analysis of the edit and Hamming distances of the indices set. a, shows the results for indices of length 11. b, shows the results for indices of length 12, including the file identifier. The histogram shows that most of the indices are at a distance of 5 or more from one another.

E.3 Diagonal Columns Encoding

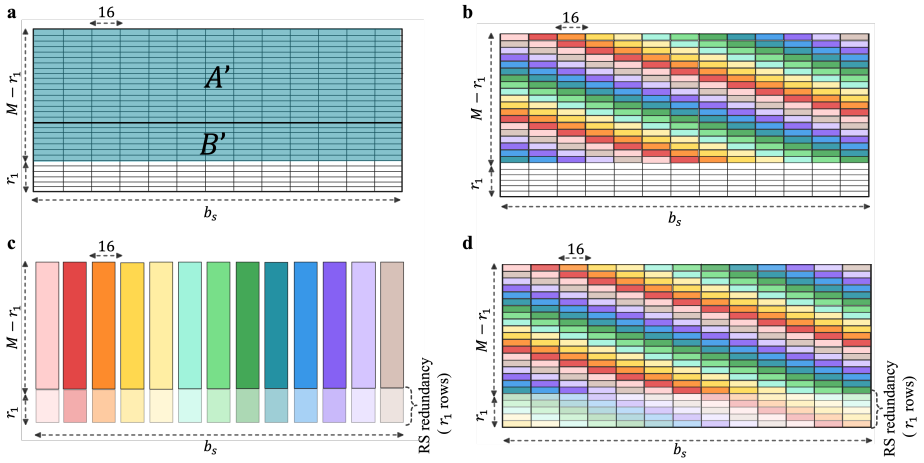
Similarly to the methods that have been presented in previous works [12, 14, 19] in step 3 of our encoding process, we encode the column of our matrix with an error-correcting code, more specifically an $(M, M-r_1, r_1+1)$ RS code. As some sequencing and synthesis technologies are more error-prone at the beginning and end of each read [39], designing the RS code to work directly on the columns of our matrix would have required more redundancy symbols in the encoding of the columns closer to the beginning and end of the encoded sequences (left and right sides of the matrix). Hence, we encode columns diagonally to allow a more uniform distribution of the errors. This allows us to use the same amount of redundancy symbols for each diagonal column. Our diagonal columns encoding is presented in Supplementary Figure E.9 and is performed as follows. Note that the figure represents the case in which we encode using constraint coding. In the non-constrained setup, there should be an additional 16-bit column that is used to encode information.

1) Given the $(M-r_1) \times b_s$ binary matrix composed of A' and B', partition the matrix into $\frac{b_s}{16}$ blocks of columns (each block composed of 16 bits) as depict in Supplementary Figure E.9.a.

2) Define $\frac{b_s}{16}$ diagonal columns by shifting the block that is taking from each row by 16 bit from the location in the previous row as depicted in Supplementary Figure E.9.b.

3) Encode each diagonal column using the encoder of the RS code as depicted in Supplementary Figure E.9.c.

4) Place the redundancy symbols according to the same diagonal shift as shown in Supplementary Figure E.9.d.



Supplementary Figure E.9: Schematic description of the diagonal-column encoding. a, partition the matrix into 16 bits width columns. b, define the diagonal columns by a cyclic shift of 16 bits between each row. c, encode each diagonal column using RS code. d, insert the redundancy from the RS code back into the appropriate diagonal column.

E.4 Constrained Code

In DNA-based storage systems, several structural constraints should be enforced on the encoded sequences. Such constraints include limiting the length of the homopolymers, avoiding the presence of specific motifs (short sequences) as substrings of the encoded sequences, and controlling the amount of G and C bases that occur in each encoded sequence. Such constraints are used to mitigate the error rates in the biological process (synthesis, PCR, and sequencing) involved in DNA-based storage systems, and the specific constraints and the parameters of these constraints should be selected according to the specific technology that is being used.

Incorporating constrained codes together with ECC requires careful attention as these two modules cannot be straightforwardly concatenated with one another and there is no general approach to how to pair the two. To overcome this challenge, we designed our constrained code to interleave with the ECC in a simple manner that allows the selection of specific design constraints. For our experiment, we focused on the constraints of limiting the length of the homopolymers to be at most 4. Additionally, we restrict the amount of G and C bases in each encoded sequence to be between 30% and 70% in the worst case and between 45% to

55% with high probability. We note that the suggested constrained coding is based on block encoding and thus it can be adapted to support additional constraints by designing the exact mapping between a block of bits to a block of DNA bases. The only thing that should be maintained to integrate this modified code into our scheme is that the decoding of each block can be done independently of the other blocks.

Our constrained code is based on a predefined mapping function that takes n_1 bits and translates them to a sequence of length n_2 over the DNA alphabet $\{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$. The encoding is done by partitioning the n_1 bits into non-overlapping blocks and encoding the blocks of bits into blocks over the DNA alphabet sequentially. If we don't want to enforce any constraints, we use the simple 2 bits to 1 symbol mapping: $00 \rightarrow \mathsf{A}, 01 \rightarrow \mathsf{C}, 10 \rightarrow \mathsf{G}, 11 \rightarrow \mathsf{T}$. Otherwise, n_1 can be either $b_s = 208$ or $b_\ell = 238$. In the first case, $n_1 = 208$ and the input is partitioned into 16 blocks of 13 bits each. In the second case, $n_1 = 238$ and the input is partitioned into 18 blocks, the first 16 are of length 13 each, and the last 2 blocks are of length 15 each. The blocks of 13 bits are encoded into DNA blocks of length 7, while the blocks of length 15 are encoded into DNA blocks of length 8. The set of allowed DNA blocks (with the corresponding length) consists of all the sequences over $\{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$ that do not contain 5 identical consecutive symbols in the middle and 3 identical consecutive symbols in the edges (which guarantee that concatenation of such blocks will never result with a sequence of 5 identical consecutive symbols).

In our coding scheme, we only consider blocks of length 7 or 8. The set of allowed DNA blocks of length 7 is partitioned into 8 groups based on the number of occurrences of G or C symbols within them (from 0 to 7 GC-content). For any binary input of length 13, the mapping either translates the input to a sequence that belongs to the group with 3 or 4 GC content termed, "almost balanced" GC words, or to a couple of sequences, with two different GC contents. The set of allowed DNA blocks of length 8 is partitioned into 9 groups based on the number of occurrences of G or C symbols within them (from 0 to 8 GC-content). For any binary input of length 15, the mapping either translates the input to a sequence that belongs to the group with 4 GC content, termed "balanced" GC words, or to a couple of sequences with two different GC contents. The complete mappings that were used in our encoder can be found in the code repository published with this work.

Next, we describe how our encoding process enforces the constraints.

1) **Limited homopolymer constraint.** As described above, each of the DNA blocks that we use in our mapping has at most two identical bases at both edges. Furthermore, these blocks have at most four identical bases in the middle. Therefore, concatenating any two blocks cannot create a homopolymer of length 5 or above. Additionally, as described in the Index encoding subsection, the first 11 bases of our selected indices do not contain homopolymers of length 5 or above, and their right-most homopolymer is of length at most 4. Thus, it can be verified that we can always select the file identifier (the 12-th symbol of the index) out of the two possible options in a way that will preserve the homopolymer constraint on the entire encoded sequence.

2) **GC-content constraint.** The encoding process of bits into DNA blocks is done sequentially, from left to right. As described in the mapping included in the code repository published with this work, most of the bit blocks are mapped into DNA blocks with either

balanced or almost balanced GC content, whereas the rest of the bit blocks are mapped to two different options of DNA blocks with a non-balanced GC content. Thus, in every step of the encoding process, if there is more than a single DNA block that can be used, the encoder selects the one that keeps the GC content closer to 50%. It can be verified that this ensures GC content of more than 30% and less than 70% (including the index) in the worst case and with high probability the GC content is more than 45% and less than 55%.

Note that the design of our coding scheme allows us to apply constrained coding on almost all of the encoded sequences in the worst case. However, a small fraction of the encoded sequences (in our case $r_3 = 4786$ encoded sequences) satisfies the constraints only with high probability. More precisely, we cannot apply our constrained coding on the redundancy part of the tensor product code (sub-matrix D in Supplementary Figure E.7.e). Meaning that, with a small probability, the suffix of the corresponding encoded sequences might contain longer homopolymers.

E.5 Tensor-Product Code

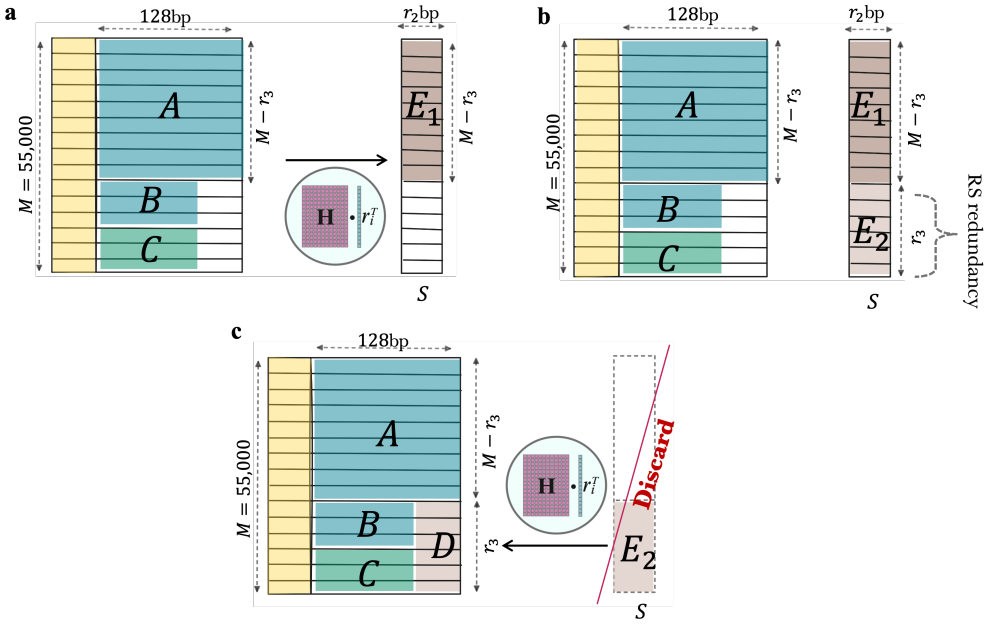
In inner-outer code approaches, there is an inner code that protects each encoded sequence that encodes the data from errors within its symbols, as well as an outer code that protects an erasure of a strand. By design, the DNAformer reconstruction eliminates most of the deletions and insertions, and the output has the same length as the encoded sequence. Hence, after this step, we need to only take care of substitution errors, which are easier to solve in terms of coding theory and require less redundancy.

For each cluster, the DNN produces a soft output that is transferred to the confidence filter, which decides whether to produce an output, activate the CPL, or ignore the cluster. The actual outputs of the DNN can be partitioned into four distinct sets:

- Correct predictions - most of the clusters (roughly 85%–100%, see Figure 8.3.a).
- Missing clusters - can be corrected by the outer code easily (requires small redundancy).
- Wrong predictions with a small number of substitutions (3 or less).
- Wrong predictions with many substitutions (more than 3).

As the size of the 4-th group is very small (in our experiments, between 0% and 3% of the clusters), it is wasteful to encode each of the sequences with an inner code that can correct many errors. Moreover, it is wasteful to encode all sequences with inner code, even for a small number of errors, as most of the predictions are correct. We use these observations to design a code for the DNN outputs and consider the confidence parameter. The key point is that by using the confidence filter, we can classify the outputs from the 4-th group (with very high accuracy) and ignore them (i.e., classifying them as missing clusters) or activate the CPL algorithm on the corresponding clusters. It should be noted that the tensor-product method can be used without the confidence of the DNN, but in that case, additional redundancy will be required.

Our encoding process utilizes a matrix $\mathbf{H}$, which is a 16×128 parity-check matrix of a BCH code over the 4-ary alphabet, that can correct 3 substitution errors. Additionally, the matrix $\mathbf{H}$ can be found in the code repository published with this work. Given the matrix $\mathbf{H}$, and the matrices A, B, and C, obtained from the previous steps of our encoding (see Supplementary Figure E.10), the TP encoding works as follows.



Supplementary Figure E.10: Description of the TP encoding. a, multiple $\mathbf{H}$ by each row of A. b, encode E_1 by an RS code. c, complete the matrix such that any row satisfies the syndrome vector equation. Following this step, we can discard S, which will be reconstructed in the decoding process.

1) Define an $M \times r_2$ empty matrix S.

2) For each of the first $M - r_3$ rows of the matrix $\mathcal{X}$ (the rows that correspond to the submatrix A), update the i-th row of S, S_i, with the vector $\mathbf{H} \times r_i^T$, where r_i is the i-th row of A. This step is presented in Supplementary Figure E.10.a.

3) Encode the first $M - r_3$ rows of S using a RS code with r_3 redundancy symbols as depicted in Supplementary Figure E.10.b.

4) To obtain the matrix X from $\mathcal{X}$, fill the $r_3 \times r_2$ bottom-right submatrix of $\mathcal{X}$, such that for each row of X, r_i for $1 \leq i \leq M$, we have that $\mathbf{H} \cdot r_i^T = S_i^T$, termed syndrome vector equation. This step is depicted in Supplementary Figure E.10.c. Following this step, we can discard S, which will be reconstructed in the decoding process.

Note that after the last step, the first $M - r_3$ rows satisfy the required property $\mathbf{H} \cdot r_i^T = S_i^T$ by the definition of the vector S. Moreover, for the remaining r_3 rows, it is possible to solve this

system of linear equations since $\mathbf{H}$ is a full-rank matrix with degree that is equal to $r_2 = 16$, and the last 16 columns are linearly independent.

E.6 Decoding

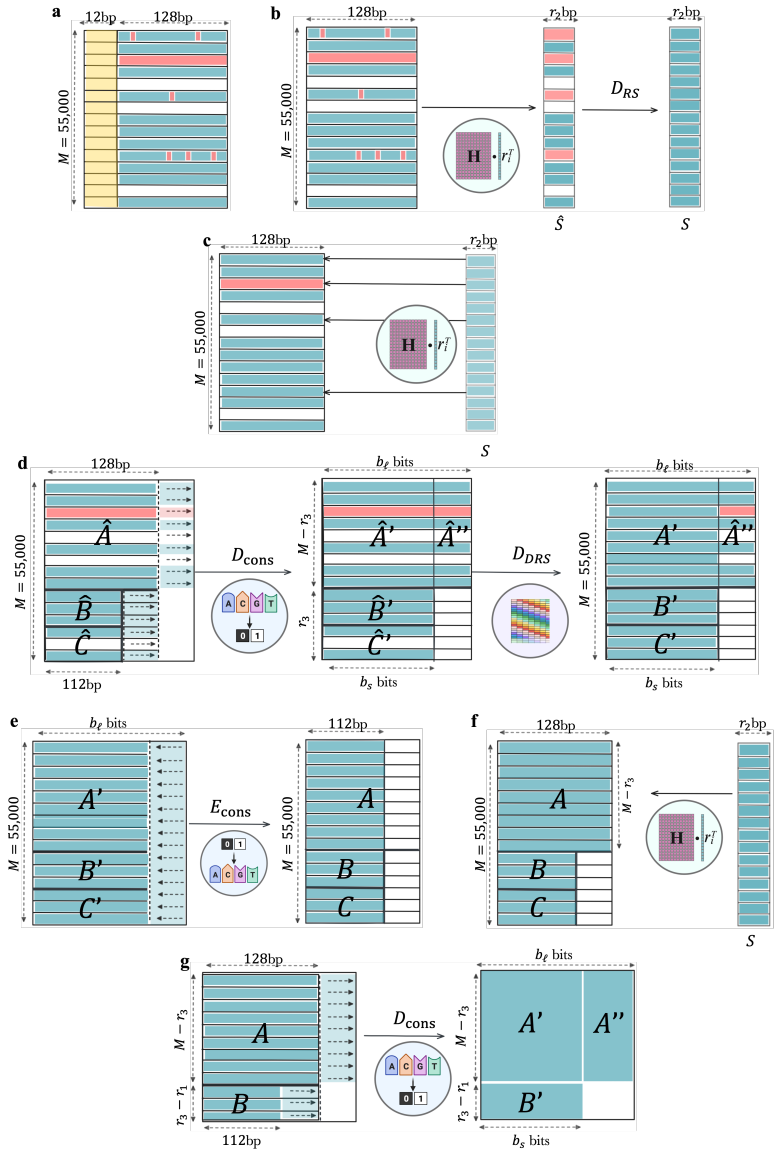
We start by organizing the predictions from the DNAformer in a matrix $\hat{X}$, with M rows, which is a noisy version of the matrix X. This can be done since our binning algorithm only considers correct indices. The matrix $\hat{X}$ is illustrated incSupplementary Figure E.11.a. Let α denote the number of rows in $\hat{X}$ with three or less substitution errors, let β denote the number of rows in $\hat{X}$ with more than three substitution errors, and let γ denote the number of missing rows in $\hat{X}$. Given $\hat{X}$, our decoding process is based on the following steps.

1) **Recover the matrix S.** The first step in our decoding algorithm is to recover the matrix S that was used during the TP encoding. To this end, we remove the 12 first symbols of each row in $\hat{X}$ (i.e. remove the index and file identifier of each row). Then we define $\hat{S}$ to be the $M \times r_2$ matrix in which the i-th row, for $1 \leq i \leq M$, is equal to $\mathbf{H} \cdot r_i^T$, where r_i is the i-th row of $\hat{X}$ (after removing the first 12 columns). Lastly, we decode the matrix $\hat{S}$ using D_{RS}, the decoder of the RS code that is used in our TP code. Note that for any i, if the prediction of DNAformer in the i-th row of $\hat{X}$ is correct, then the i-th row of $\hat{S}$ is equal to the i-th row of S. This implies that $\hat{S}$ is a noisy version of S, with at most $\alpha + \beta$ wrong rows and γ missing rows. Hence, for any three positive integers α, β, γ such that $2(\alpha + \beta) + \gamma \leq r_3$, we have that $D_{RS}(\hat{S}) = S$, i.e., D_{RS} guarantees successful recover of S from $\hat{S}$. This process is illustrated in Supplementary Figure E.11.b.

2) **Correct rows with up to 3 errors**. The next step is to correct the α rows in $\hat{X}$ that have 3 or less errors. This is done by first identifying the rows that are different in S and $\hat{S}$. Then for each such row i, the i-th row of $\hat{X}$ is decoded using the parity-check matrix $\mathbf{H}$ and the i-th row of S and $\hat{S}$. Since $\mathbf{H}$ is a parity-check matrix of a BCH code that can correct up to 3 substitutions, each of the α rows with 3 or less errors will be corrected after this step, as depicted incSupplementary Figure E.11.c.

3) **Recover the submatrices A', B', and C'.** In this part of the decoding, we ignore the submatrix of $\hat{X}$ that corresponds to D and only consider the submatrices $\hat{A}, \hat{B}$, and $\hat{C}$, which are noisy versions of the submatrices A, B, and C. We first decode each 4-ary row (of the submatrices $\hat{A}, \hat{B}$, and $\hat{C}$) into a row of bits using D_{Cons}, the decoder of our constrained code. Then we decode the columns of $\hat{A}', \hat{B}'$, and $\hat{C}'$ using D_{DRS} the decoder of our diagonal RS code. Since any correct row in $\hat{A}, \hat{B}, \hat{C}$ will result in a correct binary row in $\hat{A}', \hat{B}', \hat{C}'$, respectively, these three submatrices together are a noisy version of the submatrix that is composed of A', B', and C', with at most β wrong rows and γ missing rows. Thus, for any two positive integers β and γ such that $2\beta + \gamma \leq r_1$, D_{DRS} successfully recovers A', B', and C' from $\hat{A}', \hat{B}', \hat{C}'$. This step in presented in Supplementary Figure E.11.d.

4) **Recover the submatrices A, B, and C.** To this end, we ignore the submatrix $\hat{A}''$ and convert A', B', and C' back to their 4-ary representation by encoding each row with



Supplementary Figure E.11: Description of the decoding step. The colormap is as follows, the correct predictions are the green rows. Substitutions are highlighted in red. There are α rows with three or less substitutions shown as green rows with few highlighted errors within them. There are β rows with more than three errors and are shown as complete red rows. There are γ rows which are the missing rows as shown as white rows. a, input to the decoder. b, reconstructing the syndrome vector S using $\mathbf{H}$ matrix. c, using S and $\mathbf{H}$ to reconstruct the α rows with up to three errors. d, transform the matrix back into binary format and use the diagonal encoder to correct A', B' and C'. e, transforming A', B', and C' back to base 4. f, reconstruct A using S and $\mathbf{H}$. g, transforming back into binary data.

E_{Cons} as presented in Supplementary Figure E.11.e. Then, we use the matrix $\mathbf{H}$ again to recover the remaining symbols of A, which can be done since the i-th row of A, should satisfy $\mathbf{H} \cdot r_i^T = S_i^T$, where S_i is the i-th row of the matrix S that we already recovered. This step is shown in Supplementary Figure E.11.f.

5) **Recover the binary data x.** Finally, as shown in Supplementary Figure E.11.g, the submatrices A and B are converted back into bits using D_{Cons} and the binary data $\boldsymbol{x}$ is obtained by concatenating the rows of the latter.

Supplementary F: Analysis of Decoder Robustness

The discussion above implies that our decoding process recovers X correctly if the following two conditions hold: $2(\alpha+\beta)+\gamma \leq r_3$, $2\beta+\gamma \leq r_1$. This relation is illustrated in Figure 8.4.b in which for any pair of parameters α and β we present the maximal γ for which our decoder is guaranteed to successfully decode $\hat{X}$ into X.

It can be observed that even for fixed values of r_1, r_2, and r_3, there is a wide range of values for α, β, and γ, for which the decoding is successful. The robustness of our decoding algorithm is rooted in the tradeoff between these three quantities. More precisely, our confidence filter was designed to replace rows with substitutions (i.e., rows that were counted as either α or β) with missing rows (i.e., counted as γ). As α and β have a factor of two in our equations (and γ doesn't), our confidence filter increases the safety margin of our entire DNA retrieval pipeline.

Additionally, when the CPL algorithm is activated, wrong predictions of the DNN can be corrected by the CPL reducing the total number of rows that are not correct in our pre-decoding matrix. In addition, the CPL can reduce the number of errors in the final DNAformer prediction which increases α while decreasing β. This improved the safety margin further as r_3 is larger than r_1.

The values of α, β, and γ for File 1 and File 2 are given in Supplementary Table E.10, where the values are presented for the Illumina test dataset and the two Nanopore test datasets. For each dataset, the three configurations of the DNAformer are considered. The first corresponds to the setup in which only the DNN is activated within the DNAformer. The second corresponds to the case in which the confidence filter is also activated. The last, which is our most robust setup, considers the configuration in which both the confidence filter and the CPL are activated within the DNAformer. The rows highlighted in red reflect cases where the decoder failed. The results show that for the Illumina data and the Nanopore two flowcells dataset, all the configurations of the DNAformer successfully retrieved the binary data. In the Nanopore single flowcell, which has the lowest SNR, the safety margin mechanism was needed to improve the DNAformer accuracy and retrieve the binary data successfully.

Supplementary G: Confidence Filter and Safety Margin

The safety margin is a metric that describes how robust the DNAformer is under specific working conditions. We derive the safety margin from the error correcting capabilities of the decoder shown in the previous section. It is defined as $\min\{r_1 - 2\beta - \gamma, r_3 - 2(\alpha+\beta) - \gamma\}$, where

		Illumina test dataset			Nanopore test dataset single flowcell			Nanopore test dataset two flowcells		
		α	β	γ	α	β	γ	α	β	γ
File 1	DNN	2	1	28	1,337	1,440	34	882	479	14
	DNN+Confidence	2	1	28	1,229	827	814	879	437	65
	DNN+Confidence+CPL	2	1	28	1,075	235	814	805	82	65
File 2	DNN	2	1	28	1,351	1,422	38	843	515	10
	DNN+Confidence	2	1	28	1,273	841	779	836	477	57
	DNN+Confidence+CPL	2	1	28	1,108	252	779	743	93	57

Supplementary Table E.10: Analysis of the different types of errors for different DNAformer configurations. The errors are shown for each of the three datasets and for each file separately. Red signifies failed retrieval. The results show that for the Illumina and Nanopore two flowcells, the DNN can retrieve the data on its own. However, for the Nanopore single flowcell the confidence filter and the CPL are needed to ensure retrieval.

negative values correspond to cases in which the DNAformer fails to retrieve the information. We recall that Supplementary Table E.10 summarizes the values of α, β, γ for our Nanopore and Illumina datasets. In the table, the cases where the safety margin is negative are highlighted in red, in these cases the DNAformer fails to retrieve the information.

In our methodology, the design for a required safety margin is achievable via two control parameters, termed $cluster\ size_{threshold}$ which filters the minimum cluster size, and $confidence_{threshold}$ which sends bad predictions of the DNN to the CPL. The optimization process of the $cluster\ size_{threshold}$ included a similar methodology as described in the 'Effects of cluster size on the error rate' section. In this process, we sampled different number of reads for each cluster from the Nanopore pilot dataset and examined the success rate, average Hamming distance, and average edit distance of the DNN's predictions. This analysis resulted in $cluster\ size_{threshold} = 4$, where the success rate dropped below 50%. Similar behavior was observed in the validation analysis performed on the Nanopore two flowcells as shown in Supplementary Figure A.1.

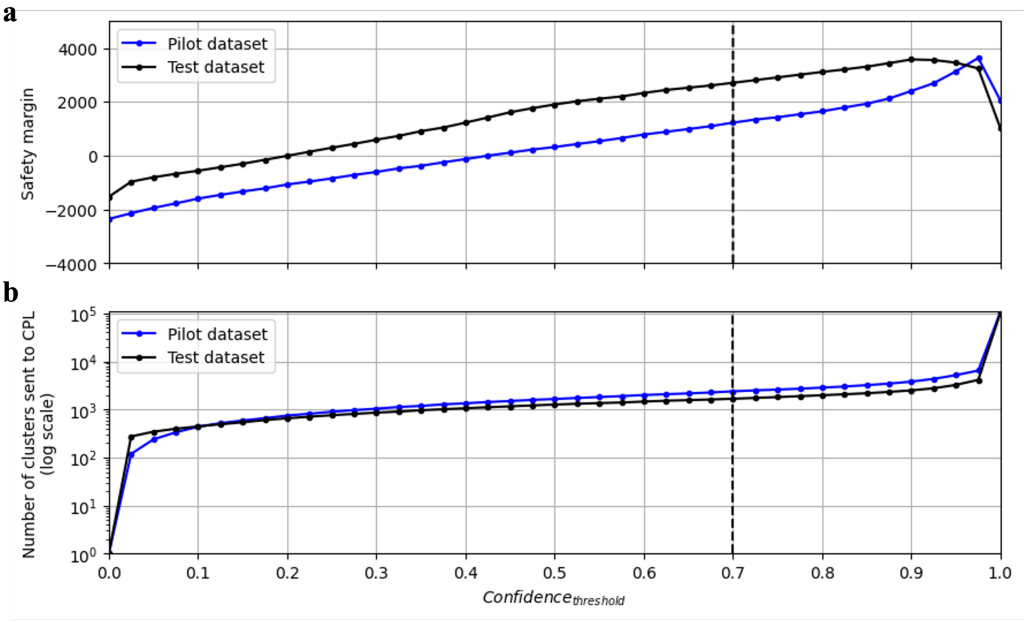
Based on our system methodology, the optimization process of the $confidence_{threshold}$ relies on an analysis of the pilot datasets. As the pilot dataset contains only $1,000$ clusters, the average cluster size of the pilot dataset is much larger compared to the one in the test datasets which consists of $110,000$ clusters. This gap in cluster size also creates a distribution that is similar in shape, but different in scale, as described in Supplementary Figure B.3. To overcome this gap and normalize the distributions, we sample 2% of the reads from the pilot dataset and then perform the binning step. This process was repeated 110 times to create a normalized pilot dataset of up to $110,000$ clusters, in which all the reads are taken from the real pilot datasets.

The optimization process of the $confidence_{threshold}$ is shown in Supplementary Figure E.12.a where we show the safety margin for the normalized Nanopore pilot. In our design, we chose a safety margin of at least 1% or $1,100$ wrong predictions which corresponds to

$$confidence_{threshold} = 0.7.$$

Note that different values of the safety margin will derive different values for the

$confidence_{threshold}$. In cases where a very tight fit between the pilot and test dataset is required, we recommend a larger pilot dataset of $5,000$–$10,000$ clusters.



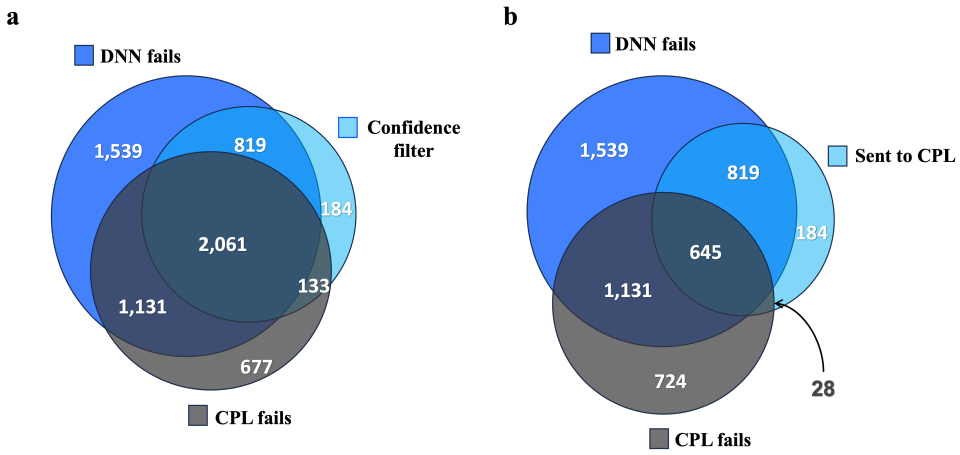
Supplementary Figure E.12: Analysis of the safety margin and confidence threshold. Results of the confidence evaluation experiment on the Nanopore test single flowcell dataset. The X-axis represents the confidence threshold a, shows the safety margin for the normalized Nanopore pilot and test datasets. In the dashed line, we show the chosen confidence threshold. b, shows the number of clusters that are sent to the CPL algorithm.

Supplementary Figure E.12.b shows the number of clusters sent to the CPL for various values of $confidence_{threshold}$. This graph illustrates the tradeoff between a larger safety margin and the total runtime of the system. As more clusters are sent to the CPL, the overall runtime increases.

The test dataset shown in Supplementary Figure E.12.a-b is the Nanopore single flowcell. This was chosen as the Illumina test dataset and Nanopore two flowcells test dataset are cases where the DNN can successfully retrieve the information without the confidence filter and CPL modules. However, the Nanopore single flowcell is a more challenging case that requires these modules for successful information retrieval.

Lastly, we provide an analysis of the effect of the confidence function on our information retrieval pipeline. The analysis involved running both the DNN and CPL on the entire Nanopore test dataset single flowcell. Supplementary Figure E.13 presents Venn diagrams of the intersection between the number of clusters that have wrong DNN predictions, the number of clusters that were sent to the CPL algorithm by the confidence filter, and the number of clusters that have wrong CPL predictions. A wrong prediction is defined as an output from the DNN or CPL with at least one wrong symbol.

The diagrams presented in Supplementary Figure E.13 show the benefits of incorporating



Supplementary Figure E.13: Detailed analysis of combining between the DNN, confidence filter, and CPL. Venn diagrams show the intersection between cases where the DNN fails, the CPL fails and the combination between them using the confidence filter to maximize performance. a, shows how the confidence filter interacts with the set of clusters for which the DNN or CPL fails. b, shows similar relations after filtering erasures.

the confidence filter and the CPL algorithm into our retrieval pipeline Supplementary Figure E.13.a considers erasures as wrong predictions while Supplementary Figure E.13.b ignores erasures. To fully understand the results, these two Venn diagrams should be considered together. The diagram in Supplementary Figure E.13.a shows that among the $5,550$ wrong predictions of the DNN, the confidence filter was able to detect $2,880$, which are described in the Venn diagram by the intersection between DNN fails and Confidence filter. The $2,880$ detected clusters comprise roughly 52% of these wrong predictions.

Note that the predictions filtered by the confidence filter correspond with three types of clusters: empty clusters, clusters that were passed to the CPL, and clusters that were screened by the confidence filter. As shown in Supplementary Figure E.13.b, among the DNN predictions, the number of clusters that were sent to the CPL is $1,676$, which is 30% of the wrong and missing predictions of the DNN. Among these $1,676$ clusters, 59% ($1,003$) were successfully corrected by CPL. In total, it means that out of all the wrong and missing predictions of the DNN, 15% were corrected by the CPL. Among the clusters that were sent to the CPL, there were 133 (4.1%), for which the DNN's outputs were correct while the CPL returned wrong predictions. For the rest of the clusters that were passed to the CPL, either the DNN and the CPL have corrected predictions, or both have wrong ones. Hence, they do not affect the final output.

To conclude, the diagrams show how incorporating the CPL into the pipeline increases the number of correct predictions and converts wrong predictions with missing ones, which increases the safety margin of our DNA retrieval pipeline.

Chapter 9

Cover Your Bases: How to Minimize the Sequencing Coverage in DNA Storage Systems

Daniella Bar-Lev, Omer Sabary, Ryan Gabrys, and Eitan Yaakobi

Abstract

Although the expenses associated with DNA sequencing have been rapidly decreasing, the current cost of sequencing information stands at roughly $120/GB, which is dramatically more expensive than reading from existing archival storage solutions today. In this work, we aim to reduce not only the cost but also the latency of DNA storage by initiating the study of the ***DNA coverage depth problem***, which aims to reduce the required number of reads to retrieve information from the storage system. Under this framework, our main goal is to understand the effect of error-correcting codes and retrieval algorithms on the required sequencing coverage depth. We establish that the expected number of reads that are required for information retrieval is minimized when the channel follows a uniform distribution. We also derive upper and lower bounds on the probability distribution of this number of required reads and provide a comprehensive upper and lower bound on its expected value. We further prove that for a noiseless channel and uniform distribution, MDS codes are optimal in terms of minimizing the expected number of reads. Additionally, we study the DNA coverage depth problem under the random-access setup, in which the user aims to retrieve just a specific information unit from the entire DNA storage system. We prove that the expected retrieval time is at least k for $[n,k]$ MDS codes as well as for other families of codes. Furthermore, we present explicit code constructions that achieve expected retrieval times below k and evaluate their performance through analytical methods and simulations. Lastly, we provide lower bounds on the maximum expected retrieval time. Our findings offer valuable insights for reducing the cost and latency of DNA storage.

9.1 Introduction

The world's digital data is growing exponentially, doubling from 30 to 64 zettabytes in just three years, and it is anticipated to reach 180 zettabytes by 2025, resulting in a data storage crisis. The demand for storage capacity already exceeds the supply, and the gap continues to grow [27]. Recent research and insights from the IDC emphasize the struggle of existing storage technologies to meet the demands of the big data era.

Recognizing this challenge, DNA emerges as a promising storage medium due to its exceptional density and durability. The DNA storage pipeline usually involves three main components. The first is *DNA synthesis*, which produces artificial DNA molecules. These synthetic DNA molecules are called *oligos* or *strands* and they can be designed in a way that encodes the user's information. The current synthesis technologies only produce strands that are up to a length of 300 bases [22] and due to technology limitations, they also produce several noisy copies per encoded strand. Thus, it is likely that the user information is stored in several different strands. The second component of the DNA storage pipeline is a *storage container*, usually a small tube that contains all the short strands that encode the user information. Lastly, to read back the user information, it is required to perform *DNA sequencing* on the strands in the tube. The sequencing process translates the DNA strands into digital sequences over the DNA alphabet, which are noisy copies of the synthesized strands. These DNA sequences can be decoded to read back the user's information.

The sequencing process, which is done using a DNA sequencer, is one of the principal components in any DNA storage system [1, 12, 24, 34]. Nowadays, DNA sequencers suffer from relatively slow throughput as well as high costs relative to other alternative storage technologies [30, 35, 37]. These issues are related to the so-called *coverage depth* of DNA storage, which is defined as the ratio between the number of reads that are sequenced and the number of designed strands [17]. Reducing the coverage depth can improve the latency of any existing DNA storage system and reduce its costs.

Motivated by the connection between the coverage depth, latency, and cost, and in an effort to design coding schemes that overcome the drawbacks associated with existing sequencing technologies, in this work we initiate the study of a novel problem, referred to as the *DNA coverage depth problem*. Simply stated, the DNA coverage depth problem aims to minimize the coverage depth while maintaining system reliability. We will study the required coverage depth as a function of the DNA storage channel, the error-correcting code, and the algorithms involved in retrieving the user's information. Furthermore, we seek to understand how to pair an error-correcting code with a given DNA storage system in order to minimize the coverage depth. This problem will be studied under both the random and non-random access settings. While the latter addresses the problem of retrieving all the information that was being stored, the former describes the case in which the user is interested in retrieving only a specific part of the stored information. Moreover, we plan to suggest coding schemes that optimize the required coverage depth and to study, both theoretically and experimentally, how one can utilize codes to minimize the sequencing time and costs.

Despite significant work on DNA storage, only a small number of works have focused on reducing the latency and costs associated with sequencing in experimental or theoretical setups. Erlich et. al. [12] encoded digital information into DNA strands using a Luby transform-

based coding scheme. Later, they diluted their synthesized strands and studied the effect of this dilution on their ability to sequence and decode the information. The dilution procedure reduced the potential (maximal) coverage depth of their system down to roughly 1300 reads per strand, thus making the decoding process more challenging. They showed that thanks to the error-correcting capability of their scheme, they were able to perfectly retrieve the stored information. In another related work, Chandak et. al. [6] defined the ratio between the number of synthesized bits and the number of information bits as the *writing cost*, and similarly the ratio between the number of bits that have to be read (sequenced) and the number of information bits was defined as the *reading cost*. In their work, they studied the tradeoffs and relations between the writing and reading costs. They first showed that for the noiseless channel, it is enough to read one copy per designed strand. Thus, the relation of these two costs can be obtained by inferring the channel as an erasure channel with an erasure probability that can be approximated using Poisson approximation. Additionally, the authors suggested an LDPC-based coding scheme that can improve the ratio between the two costs. They also showed by simulations how their suggested scheme can be used with different redundancy levels to reduce both the writing cost and the reading cost.

The DNA coverage depth problem is related to the *coupon collector's (CCP)*, *dixie cup*, and *urn* problems [11,14,15,23]. For all these problems, it is assumed that there are n different types of coupons and the question of interest is *how many coupons one should collect before possessing one coupon of each type*. It is well known that if the coupons are drawn uniformly at random (with repetition), then the expected number of coupons necessary to have at least one coupon from each type is roughly $n \log n$. Under our setting, the coupons refer to the copies of the synthesized oligos and the goal is to read at least one copy of every oligo.

The CCP has several generalizations [11, 15, 23], some of which will be explored in this work. One such problem, which is referred to as the *MDS coverage depth* problem, is *how many coupons one should collect before possessing t copies of k coupons*. This generalization represents the scenario where a reconstruction algorithm that requires t reads of an oligo for successful decoding is used along with an MDS code that requires correctly retrieving k out of the n synthesized sequences to recover the stored encoded information. Another problem that is addressed in this paper is the *coding coverage depth problem*, which generalizes the MDS coverage depth and considers the effect of an error-correcting code, which is not necessarily an MDS code. Under this setup, our main results show that MDS codes are optimal codes for the purpose of reducing the expected coverage depth. Furthermore, our analysis for the MDS coverage depth problem provides a deep understanding of the required number of reads that should be sequenced in order to guarantee a successful retrieval of the information with high probability.

Additionally, motivated by the random-access setting where one wishes to retrieve a single strand of DNA from a storage system, in Section 9.6 we consider another problem that is related to the CCP, but to the best of our knowledge has not been studied before. Suppose we are given k information coupons which we can encode into a set of n total coupons. *For any information coupon say i, what is the expected number of coupons that need to be collected in order to retrieve the information in coupon i?* Trivially, if no code is used and every coupon is collected with the same probability, then the expected number of coupons that need to be collected is equal to k. In Section 9.6, we initiate the study of this problem, which we refer to

as the *singleton-random-access problem*. Our main result is to show that it is indeed possible to design coding schemes that allow random access that requires less than k coupons and provide examples of several such schemes.

This paper is organized as follows. Section 9.2 introduces the definitions that are used throughout the paper. In Section 9.3, we formally define the problems that are studied throughout this paper along with related work. Section 9.3 also gives a detailed summary of the main results given in this paper. Next, in Section 9.4, we consider the case in which the channel is noiseless and address the MDS coverage depth problem and the coding coverage depth problem for the noiseless channel. Section 9.5 extends the study of the MDS coverage depth problem to noisy channels, and gives several bounds on the success probability of the decoding as a function of the number of reads that were sequenced. Finally, in Section 9.6, we present our results for the singleton-random-access problem. For more details on the results and contributions presented in each section, the reader is referred to Subsection 9.3.3.

9.2 Definitions and Channel Model

In the typical model of DNA-based storage systems [12, 24, 34], the data is stored as a codeword that can be described by a vector of length-ℓ *sequences* or *strands* over the alphabet $\Sigma = \{A, C, G, T\}$. The set of all length-ℓ vectors over Σ is denoted by Σ^ℓ, and $\Sigma^* \triangleq \bigcup_{\ell=0}^{\infty} \Sigma^\ell$. For a positive integer n, $[n]$ denotes the set $\{1, \ldots, n\}$. In many cases an *outer error-correcting* $\mathcal{C}$ is used to encode the data over the length-ℓ sequences, so it is assumed that the outer code $\mathcal{C}$ receives a vector of k length-ℓ sequences, $\mathbf{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k) \in (\Sigma^\ell)^k$ and returns a vector of n length-ℓ sequences $\mathbf{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) \in (\Sigma^\ell)^n$. For two vectors $\mathbf{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{k_1})$ and $\mathbf{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{k_2})$, we denote by $\mathbf{U} \circ \mathbf{V}$ their *concatenated vector*, i.e., $\mathbf{U} \circ \mathbf{V} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{k_1}, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{k_2})$. In this work, the code $\mathcal{C}$ is denoted by (n, k) or by $[n, k]$ in case $\mathcal{C}$ is an MDS code. The vector $\mathbf{X}$ is the input to the DNA storage system, which we now describe in more detail and is also illustrated in Figure 9.1.

The DNA storage channel, denoted by S, first produces many noisy copies for each of the strands in the vector $\mathbf{X}$. Then, these noisy copies are amplified using PCR, and lastly, a *sample* of M of these strands is sequenced using a DNA sequencing technology [17]. Therefore, the output of the DNA storage channel can be described as a multiset $\mathcal{Y}_M = \{\!\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_M\}\!\}$, where each $\boldsymbol{y}_j \in \Sigma^*$ for $j \in [M]$ is called a *read* and is a noisy version of some $\boldsymbol{x}_i$, $i \in [n]$. It should be noted that our model assumes that for any read $\boldsymbol{y}_j$, the index $i \in [n]$ such that $\boldsymbol{y}_j$ is a noisy copy of $\boldsymbol{x}_i$ is known (this can be achieved by encoding the index i within the strand $\boldsymbol{x}_i$). Depending on the specific sequencing technology being used, the reads in $\mathcal{Y}_M$ can be obtained either sequentially (one after the other), or altogether. The former corresponds to Nanopore sequencing technologies [32], while the latter describes next-generation sequencing (NGS) technologies [5] (e.g. Illumina). The number of reads in $\mathcal{Y}_M$ that are noisy copies of the i-th strand $\boldsymbol{x}_i, i \in [n]$, depends upon some categorical probability distribution $\boldsymbol{p} = (p_1, \ldots, p_n)$, where for $i \in [n]$, p_i is the probability to sample a read of $\boldsymbol{x}_i$. The probability distribution $\boldsymbol{p}$ is a function of the DNA storage channel S and is referred by the *channel probability distribution*, or in short *channel distribution*; Note that the distribution $\boldsymbol{p}$ might also depend on the design of the DNA strands in $\mathbf{X}$, however for simplicity, in this work we assume that $\boldsymbol{p}$ is only a function

of the channel S. Moreover, it is assumed that $p_i > 0$ for all $i \in [n]$, unless stated otherwise.

Remark 9.1. *Note that in several works, see e.g. [21, 29], it is assumed that a* set *(and not a vector) of strands is stored in the DNA storage system. However, since the strands in these sets are tagged by indices anyway, we assume for simplicity that the information is a vector of strands. Furthermore, it may also be possible that every strand is encoded using an inner code [12, 24]. Nevertheless, since this part is independent of the study of this work, it is not treated as part of the encoding process, but it is taken into account in the success probability of a retrieval algorithm, as will be explained below.*

The decoding process of $\mathbf{X}$ (and thus $\mathbf{U}$) starts with partitioning the reads in $\mathcal{Y}_M$ into groups, also called *clusters*, according to their origin strand, i.e., for $i \in [n]$, the i-th cluster should contain all the reads $\boldsymbol{y}_j$ that are noisy copies of $\boldsymbol{x}_i$. To simplify the analysis, we assume that this step is accomplished error-free. In practice, this assumption can be reached using indices in the sequence $\boldsymbol{x}_i$ which can be further protected using some error-correcting code [34]. Hence, the probability of successfully retrieving $\mathbf{X}$ and $\mathbf{U}$ mainly depends on the following two components of the solution being used.

1) *Error-correcting code*. When $\mathbf{X}$ is a codeword in some error-correcting code $\mathcal{C}$, it is possible to successfully retrieve $\mathbf{X}$ even if not all of its n symbols were decoded successfully. The applicable subsets $J \subseteq [n]$ such that $\mathbf{X}$ can be retrieved from the symbols $\boldsymbol{x}_j$ for $j \in J$ are determined by the code $\mathcal{C}$. For example, if $\mathcal{C}$ is an $[n, k]$ MDS code, then any k strands (symbols of $\mathbf{X}$) are sufficient to decode the data.

2) *The retrieval algorithm*. The success probability to retrieve the strand $\boldsymbol{x}_i$ also depends on the *retrieval algorithm*, which aims to decode a sequence using several noisy copies [4]. Typically, this probability depends on the number of noisy copies which are given as input, the channel error rates, and the use of an inner code within the strands. In this work, we model the retrieval algorithm using an integer $t \geq 1$, and we assume that each strand $\boldsymbol{x}_i$ can be retrieved given t reads, which are noisy copies of it, and cannot be retrieved given less than t reads[1].

The main goal of this paper is to study the required sample size M that guarantees successful decoding of the information. According to our model, this sample size depends on the channel, the error-correcting code, and the channel probability distribution $\boldsymbol{p}$.

Remark 9.2. *The analysis presented in this work assumes that the reads in the multiset $\mathcal{Y}_M$ are received sequentially from the DNA storage channel as illustrated in step 5a of Figure 9.1. However, our results are also relevant for the case in which all the reads are obtained together. More specifically, the random variable that governs the sample size M for which decoding is possible in the sequential case can be used to describe the non-sequential case as well. That is, the probability distribution of the latter corresponds to the decoding success probability given M strands in the non-sequential case.*

[1]Note that, in practice, the probability that the retrieval algorithm succeeds is not binary. More precisely it is a function that returns a value between 0 and 1 and increases with t.

In this paper, we explore two different scenarios concerning our problem. In the first scenario, discussed in both Section 9.4 and Section 9.5, we focus on the objective of recovering all the stored information. This involves retrieving the entire vector $\mathbf{U}$. On the other hand, in Section 9.6, we shift our attention to a different scenario where our goal is to retrieve a specific part of the information, i.e., a specific subset of symbols from the vector $\mathbf{U}$. For these scenarios, we calculate the expected required sample size for noiseless/noisy channels and study how it can be minimized using coding schemes.

9.3 The Coverage Depth Problem in the DNA Storage Channel

9.3.1 Problems Definition

This work studies the required sample size to retrieve the information vector $\mathbf{U}$, or a specific subset of its symbols, as a function of the DNA storage channel, the error-correcting code, and the retrieval algorithm. Under this framework, our goal is to understand how to optimally pair an error-correcting code with a given retrieval algorithm in order to minimize the sample size, while guaranteeing successful decoding with high probability.

According to our model characterization, we let $\nu_t^{\boldsymbol{p}}(\mathcal{C})$ be the random variable that governs the number of reads that should be sampled for successful decoding of $\mathbf{U}$. When $\mathcal{C}$ is an $[n,k]$ MDS code, this notation is replaced by $\nu_t^{\boldsymbol{p}}(n,k)$. The uniform distribution is denoted by $\boldsymbol{p}_u \triangleq (\frac{1}{n},\ldots,\frac{1}{n})$ and for brevity, we let $\nu_t(\mathcal{C}) \triangleq \nu_t^{\boldsymbol{p}_u}(\mathcal{C})$ and $\nu_t(n,k) \triangleq \nu_t^{\boldsymbol{p}_u}(n,k)$. The first two problems, which focus on retrieving the entire information vector $\mathbf{U}$, are defined below.

Problem 9.1. ***(The MDS coverage depth problem.)*** *For given values of k and n, and a channel distribution $\boldsymbol{p}$ find the expectation and the probability distribution of the random variable $\nu_t^{\boldsymbol{p}}(n,k)$. That is, find the values of $\mathbb{E}\big[\nu_t^{\boldsymbol{p}}(n,k)\big]$ and $P[\nu_t^{\boldsymbol{p}}(n,k) > m]$ for any $m \in \mathbb{N}$.*

Problem 9.2. ***(The coding coverage depth problem.)*** *For a given value of k, find the following.*

1) *Given n and $\boldsymbol{p}$, find an (n,k) code $\mathcal{C}$ that is optimal with respect to minimizing $\mathbb{E}\big[\nu_t^{\boldsymbol{p}}(\mathcal{C})\big]$.*
2) *The minimum value of $\mathbb{E}\big[\nu_t^{\boldsymbol{p}}(\mathcal{C})\big]$ over all possible codes $\mathcal{C}$ with dimension k and channel distributions $\boldsymbol{p}$. That is, find the value $\mathsf{M}^{opt}(k) \triangleq \liminf_{\mathcal{C},\boldsymbol{p}}\{\mathbb{E}\big[\nu_t^{\boldsymbol{p}}(\mathcal{C})\big]\}$.*

The third problem is related to the other setup, in which the user wishes to retrieve a subset of the k information strands (i.e., a subset of $\mathbf{U}$'s symbols). This subset can be described by an index set $I \subseteq [k]$, such that the set of information strands to be retrieved is $\mathbf{U}_I = \{\boldsymbol{u}_i : i \in I\}$. In this work, we consider the special case in which this subset is a singleton, i.e., the case where the user wishes to retrieve a single information strand $\boldsymbol{u}_i$ for some $i \in [k]$. More formally, we are interested in the following problem.

Problem 9.3. ***(The singleton coverage depth problem.)***

Given an (n,k) code $\mathcal{C}$, for $i \in [k]$, let $\tau_i(\mathcal{C})$ be the random variable that governs the number of samples to recover the i-th information strand assuming noiseless channel with uniform distribution. Find the following:

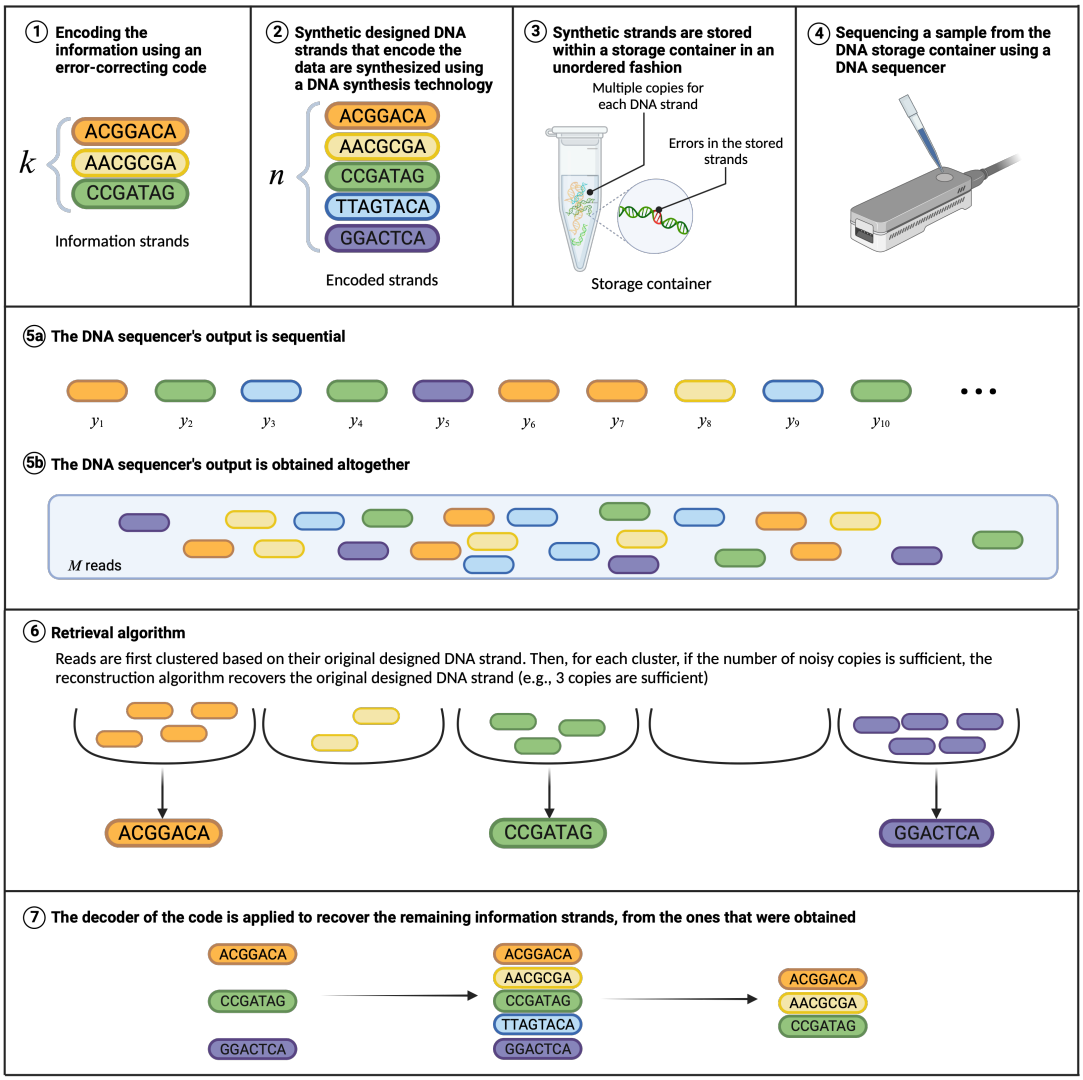


Figure 9.1: The DNA storage pipeline.

1) The expectation value $\mathbb{E}[\tau_i(\mathcal{C})]$ and the probability distribution $P[\tau_i(\mathcal{C}) > r]$ for any $r \in \mathbb{N}$.

2) The maximal expected number of samples to retrieve an information strand, i.e.,

$$T^{\mathcal{C}}_{\max} \triangleq \max_{1\le i\le k} \mathbb{E}[\tau_i(\mathcal{C})].$$

3) The average expected number of trials to retrieve an information strand, i.e.,

$$T^{\mathcal{C}}_{avg} \triangleq \frac{1}{k}\sum_{i=1}^{k} \mathbb{E}[\tau_i(\mathcal{C})].$$

When no coding is used, (i.e., $\mathbf{U} = \mathbf{X}$) $\mathcal{C}$ is removed from the notations.

9.3.2 Related Work

For the noiseless channel, it is sufficient to have a single read of each $\boldsymbol{x}_i, i \in [n]$ to retrieve it. We note that if the channel distribution is the uniform distribution $\boldsymbol{p}_u$, and no code is defined on the data (i.e., $k = n$) then finding the expectation listed in Problem 9.1 is equivalent to the classical *coupon collector's problem* [14]. This problem was first studied by Feller [14] where it was referred to as the *dixie cup problem*. Under the assumption that we have n coupons and it is equally likely to collect any of the coupons, the expected number of draws (i.e., sample size) required to get a single copy for each coupon is $\mathbb{E}[\nu_1(n, k = n)] = nH_n = n\log n + \gamma n + \mathcal{O}(1)$, where H_n is the n-th harmonic number and $\gamma \sim 0.577$ is the Euler–Mascheroni constant. Furthermore, it was also proven [15] that $\mathbb{E}[\nu_1(n,k)] = n(H_n - H_{n-k})$. It is well-known that when $\lim_{n\to\infty} n - k = \infty$,[2] the expectation can be approximated by $\mathbb{E}[\nu_1(n,k)] \approx n\log(n) - n\log(n-k) = n\log(\frac{n}{n-k})$.

For noisy channels, i.e., $t > 1$, the problem is closely related to the classical *urn problem* [11, 23]. Suppose there are n labeled urns and each can be filled with identical balls. At every round, a ball is thrown into one of the urns randomly. In each round, the probability of throwing a ball to the j-th urn is denoted by p_j, for $1 \le j \le n$, and we let $\boldsymbol{p} = (p_1, \ldots, p_n)$. In [23], it was shown that in order to have t balls in each urn (or equivalently t copies per coupon), the expected sample size is $\mathbb{E}[\nu_t(n, k = n)] = n\log n + n(t-1)\log\log n + nC_t + o(n)$, where C_t is a constant that depends on t. Following that, Erdős and Rényi [11] proved that the distribution of this random variable is tightly concentrated around the expectation. More specifically, when drawing $n\log n + n(t-1)\log\log n + nx$ times, the probability to have at least t copies for n coupons is asymptotically equal to $e^{-\frac{e^{-x}}{(t-1)!}}$ for n large enough. Flajolet et al. [15] generalized these results to a general discrete distribution on the coupons/balls and proved that the expected sample size to have at least t copies/balls for k out of the n coupons/urns is

$$\mathbb{E}[\nu^{\boldsymbol{p}}_t(n,k)] = \sum_{q=0}^{k-1} \int_0^{\infty} [u^q] \prod_{i=1}^{n} (e_{t-1}(p_i v) + u(e^{p_i v} - e_{t-1}(p_i v))) e^{-v} dv, \tag{9.1}$$

[2] In this case, there exists $0 < a < 1$, such that for n large enough $k < an$.

where $e_t(x) = \sum_{i=0}^{t} \frac{x^i}{i!}$ and for a polynomial $Q(u)$, $[u^q]Q(u)$ is the coefficient of u^q in $Q(u)$. This known result solves the expectation value listed in Problem 9.1, not only for $\boldsymbol{p}_u$ but for any channel distribution. As can be seen, for practical purposes, the expression in (9.1) and its asymptotic behavior are not easy to calculate or to work with. Hence, in Section 9.5 we solve a closely related problem and present a closed-form expression. Moreover, to the best of our knowledge, the other part of Problem 9.1, i.e., studying the cumulative probability distribution $P[\nu_t^{\boldsymbol{p}}(n,k) > m]$, is still open.

Another related problem was presented in [6] by Chandak et al. In their paper, the authors defined the *writing cost* as the number of synthesized bases per information bit, and the *reading cost* as the number of bases that have to be sequenced per information bit in order to retrieve the stored information. Their paper studies the tradeoffs between these two costs. They first showed that for the noiseless channels, the event of obtaining zero copies of a specific strand is equivalent to an erasure of this strand, which can be approximated as a Poisson random variable. Thus, they were able to compute the capacity of this channel and by this obtaining the tradeoffs of the costs. For the noisy channel, the authors suggested an LDPC-based scheme to improve the ratio between the costs, for more details see [6].

The problem of random access in DNA storage has already been addressed in several works; see e.g. [2, 20, 24, 33, 36]. The main goal is to support random access to specific DNA strands in the storage and this can be supported by the use of different primers for the different strands or physically storing strands in different storage containers. However, these solutions incur high costs, and thus the problem of storing strands together using these primers is still important and this work addresses it from a coding theory perspective.

9.3.3 Main Contributions

In this paper, we define a new family of problems that should be considered when designing DNA storage systems. Additionally, this work provides an extensive analysis and present results that enhance our understanding of the interplay between error-correcting codes, retrieval algorithms, and coverage depth. The main results with respect to each of the three problems we defined are listed below.

The MDS coverage depth problem (Problem 9.1) For this problem, we have the following results.

1) We show in Theorem 9.5 that the value of $\mathbb{E}[\nu_t^{\boldsymbol{p}}(n,k)]$ is minimized if and only if the channel has the uniform distribution.

2) We show in Theorem 9.8 and in Theorem 9.9 two upper bounds on the probability distribution of $P[\nu_t(n,k) > m]$. We further prove in Lemma 9.10 a lower bound on the probability $P[\nu_t(n,k) \leq m]$. Combining these results in Theorem 9.11 we prove that for any $\varepsilon > 0$,

$$\log\left(\frac{1}{1-R}\right) + f_c(n,R) \leq \mathbb{E}\left[\frac{\nu_t(n,k)}{n}\right] \leq \left(\log\left(\frac{1}{1-R}\right) + t\log\log n + 2\log(t+1)\right)\cdot(1+2\varepsilon),$$

where $f_c(n,R) = \mathcal{O}(\frac{1}{n^2})$.

3) For practical purposes of DNA storage systems, it is sometimes required to plan ahead and set the number of reads that should be sampled to guarantee successful decoding. Hence, we show in Theorem 9.13, that when sampling more than $r_E(n,k,t)$ reads, the expected number of encoded strands that cannot be recovered (i.e., have less than t copies) is at most $n-k$, which indicates on the probability of successful decoding. The value of $r_E(n,k,t)$ can be found in equation (9.11).

The coding coverage depth problem (Problem 9.2) We fully solve Problem 9.2 for the noiseless channel with uniform distribution. We show that MDS codes are optimal with respect to minimizing $\mathbb{E}[\nu_t(n,k)]$. We also show that for a fixed k, the larger n is, the smaller the value of $\mathbb{E}[\nu_t(n,k)]$ is. The results of this problem are given in Corollary 9.6 and Theorem 9.7.

The singleton coverage depth problem (Problem 9.3) We extensively study the singleton coverage depth problem for the case in which the channel is noiseless. Our main results are summarized below.

1) We first study and fully solve the case in which $n = k$. In particular, we prove that if $n = k$ then the expected time to retrieve a singleton is minimized when no coding is used and it is equal to k and $T^{\mathcal{C}}_{\max} = T^{\mathcal{C}}_{\text{avg}} = k$ (see Lemma 9.15 and Claim 9.18).

2) Next, to study more involved cases, we define *retrieval sets* and *minimal retrieval sets*, which correspond to the (minimal) sets of encoded strands from which a specific target singleton information strand can be recovered. Using this property of codes, we analyze the expected time to retrieve a singleton given that its minimal retrieval sets are disjoint. See these results in Theorem 9.20 and Corollary 9.22. Moreover, in Corollary 9.21 we use Theorem 9.20 to conclude that the expected time to retrieve a singleton given that $\mathcal{C}$ is the simple parity $[k+1,k]$ code is k.

3) We extend the result of Corollary 9.21 to any systematic $[n,k]$ MDS code, by the construction and detailed evaluation of the corresponding generating function. That is, we show in Theorem 9.23, that for any $[n,k]$ MDS code $\mathcal{C}$ and any $i \in [k]$, $\mathbb{E}[\tau_i(\mathcal{C})] = T^{\mathcal{C}}_{\max} = T^{\mathcal{C}}_{\text{avg}} = k$.

4) We give two explicit code constructions (Construction 9.1 and Construction 9.2) for codes with k information strands for which $T^{\mathcal{C}}_{\max} < k$, i.e., $\mathbb{E}[\tau_i(\mathcal{C})] < k$ for all $i \in [k]$. Furthermore, we analyze the behavior of these codes both analytically and by computer simulations.

5) To conclude the analysis of the singleton coverage depth problem, we provide in Lemma 9.32 and in Theorem 9.33 two lower bounds on the value of $T^{\mathcal{C}}_{\max}$. Moreover, in Corollary 9.34, for n large enough, we show that for any (n,k) code $\mathcal{C}$, such that $R = \frac{k}{n}$, we have that $T^{\mathcal{C}}_{\max} \geq k\left(\frac{1}{R} + \frac{1-R}{R^2} \cdot \ln(1-R)\right)$. In particular, the latter implies that when R approaches zero, $T^{\mathcal{C}}_{\max} \geq \frac{k}{2}$, and when R approaches one, the lower bound approaches k from below.

9.4 The Coding Coverage Depth Problem - Noiseless Channel

In this section, we focus on the setup where the channel is noiseless which refers to $t = 1$. Hence, we can assume that the retrieval algorithm simply returns the sampled sequences and thus if $\boldsymbol{x}_i$ has at least one copy, i.e., $t \geq 1$, it is enough to retrieve it. Under this setup, the minimum sample size M is equivalent to the quantity which is governed by the random variable $\nu_1^{\boldsymbol{p}}(n,k)$. Using our notation, note that the expected value of $\nu_1^{\boldsymbol{p}}(n,k)$ is given in (9.1). In this case, the distribution probability function for $\boldsymbol{p} = \boldsymbol{p}_u$ was studied in [11]. Clearly, when $k = 1$, we have that $\mathbb{E}[\nu_1(n,1)] = 1$. Hence, this section is focused on the case where $k \geq 2$. Our main result is to solve Problem 9.2 and to show that MDS codes are optimal for any categorical channel distribution. Furthermore, we show that $\mathbb{E}[\nu_1^{\boldsymbol{p}}(n,k)]$ is minimized when $\boldsymbol{p}$ is the uniform distribution and is bounded from below by $k \log e$ if $\frac{k}{n} = \Theta(1)$.

In light of the existing results and as a first step toward obtaining Theorem 9.7, we first show that for the uniform channel distribution, when k is fixed, $\mathbb{E}[\nu_1(n,k)]$ decreases as n increases.

Claim 9.3. *For all $n \geq k$, $\mathbb{E}[\nu_1(n,k)] > \mathbb{E}[\nu_1(n+1,k)]$.*

Proof. The proof follows by showing that $\mathbb{E}[\nu_1(n,k)]$ is a monotonic function that decreases with n. From [15] for any $n \in \mathbb{N}$ we have that

$$\begin{aligned}\mathbb{E}[\nu_1(n,k)] - \mathbb{E}[\nu_1(n+1,k)] &= \sum_{i=0}^{k-1} \frac{n}{n-i} - \sum_{i=0}^{k-1} \frac{n+1}{n+1-i} \\ &= \sum_{i=0}^{k-1} \left(\frac{n}{n-i} - \frac{n+1}{n+1-i} \right) = \sum_{i=0}^{k-1} \frac{i}{(n-i)(n+1-i)} > 0,\end{aligned}$$

which completes the proof. □

The next claim solves Problem 9.2.1 and states that given k information strands, for any channel distribution $\boldsymbol{p}$, using an $[n,k]$ MDS code minimizes the expectation of $\nu_1^{\boldsymbol{p}}(\mathcal{C})$ compared to any other length-n codes. This can be verified by showing that the number of subsets of size k which are sufficient to retrieve the information is maximized when an MDS code is used.

Claim 9.4. *Given k, n, and $\boldsymbol{p}$, assume that $\mathcal{C}$ is an (n,k) code. Then, it holds that, $\mathbb{E}[\nu_1^{\boldsymbol{p}}(n,k)] \leq \mathbb{E}[\nu_1^{\boldsymbol{p}}(\mathcal{C})]$, where equality is obtained if and only if $\mathcal{C}$ is an MDS code.*

Proof. Given a sample of size M, we denote by $J \subseteq [n]$ the indices of the unique encoded strands that are represented in this sample. If $|J| < k$ then it is impossible to successfully decode the information, which follows since the dimension of the code is k. Otherwise, when $|J| \geq k$, any $[n,k]$ MDS code can decode the stored information, while if $\mathcal{C}$ is not an MDS code there exists J' of size k from which the stored information cannot be decoded using $\mathcal{C}$. Therefore, if $\mathcal{C}$ is not an MDS code, for any $J \subseteq [n]$, if the information can be decoded using $\mathcal{C}$ then it can also be decoded by an $[n,k]$ MDS code. This implies the inequality stated in the theorem, where equality holds if and only if $\mathcal{C}$ is an MDS code. □

We continue towards solving Problem 9.2.2, and in the next theorem it is shown that for MDS codes, $\mathbb{E}[\nu_1^{\boldsymbol{p}}(n,k)]$ is minimized when $\boldsymbol{p} = \boldsymbol{p}_u$.

Theorem 9.5. *For any* $\boldsymbol{p}$, $\mathbb{E}[\nu_1^{\boldsymbol{p}}(n,k)] \geq \mathbb{E}[\nu_1(n,k)]$.

Proof. By (9.1), which was proven originally in [15], we have that

$$\begin{aligned}
\mathbb{E}[\nu_1^{\boldsymbol{p}}(n,k)] &= \sum_{q=0}^{k-1} \int_0^\infty [u^q] \prod_{i=1}^{n} (1 + u(e^{p_i v} - 1)) e^{-v} dv \\
&= \sum_{q=0}^{k-1} \int_0^\infty e^{-nv} \left(\sum_{\substack{I \subseteq [n] \\ |I|=q}} \prod_{i \in I} (e^{p_i v} - 1) \right) dv \\
&= \int_0^\infty e^{-nv} \cdot \sum_{q=0}^{k-1} \left(\sum_{\substack{I \subseteq [n] \\ |I|=q}} \prod_{i \in I} (e^{p_i v} - 1) \right) dv. \qquad (9.2)
\end{aligned}$$

Define $f(p_1, \ldots, p_n) \triangleq \sum_{q=0}^{k-1} \left(\sum_{\substack{I \subseteq [n] \\ |I|=q}} \prod_{i \in I} (e^{p_i v} - 1) \right)$. We show next that f is minimized if and only if $p_i = \frac{1}{n}$ for all $1 \leq i \leq n$. Furthermore, since f is minimized if and only if $\mathbb{E}[\nu_1^{\boldsymbol{p}}(n,k)]$ is minimized, this concludes the proof.

Define $g(\boldsymbol{p}) = -1 + \sum_{i=1}^{n} p_i$. Using Lagrange multipliers, the Lagrangian function is

$$\mathcal{L}(\boldsymbol{p}, \lambda) = f(\boldsymbol{p}) + \lambda g(\boldsymbol{p}) = \sum_{q=0}^{k-1} \left(\sum_{\substack{I \subseteq [n] \\ |I|=q}} \prod_{i \in I} (e^{p_i v} - 1) \right) - \lambda + \lambda \sum_{i=1}^{n} p_i.$$

We are looking for values of $\boldsymbol{p}$, that satisfy

$$\frac{\partial \mathcal{L}(\boldsymbol{p}, \lambda)}{\partial \lambda} = -1 + \sum_{i=1}^{n} p_i = 0, \qquad (9.3)$$

and for all $1 \leq i \leq n$,

$$\frac{\partial \mathcal{L}(\boldsymbol{p}, \lambda)}{\partial p_i} = \lambda + \sum_{q=1}^{k-1} \left(\sum_{\substack{I \subseteq [n] \setminus \{i\} \\ |I|=q-1}} v e^{p_i v} \prod_{j \in I} (e^{p_j v} - 1) \right) = 0,$$

which is equivalent to

$$\lambda = -v e^{p_i v} \sum_{\substack{I \subseteq [n] \setminus \{i\} \\ |I|<k-1}} \prod_{j \in I} (e^{p_j v} - 1). \qquad (9.4)$$

Hence, for any $1 \leq i < i' \leq n$ we have that

$$e^{p_i v} \sum_{\substack{I \subseteq [n] \setminus \{i\} \\ |I| < k-1}} \prod_{j \in I} (e^{p_j v} - 1) = e^{p_{i'} v} \sum_{\substack{I \subseteq [n] \setminus \{i'\} \\ |I| < k-1}} \prod_{j \in I} (e^{p_j v} - 1).$$

By reorganizing the latter equation, we have that for any $1 \leq i < i' \leq n$,

$$(e^{p_i v} - e^{p_{i'} v}) \sum_{\substack{I \subseteq [n] \setminus \{i,i'\} \\ |I| < k-1}} \prod_{j \in I} (e^{p_j v} - 1) = (e^{p_i v} - e^{p_{i'} v}) \sum_{\substack{I \subseteq [n] \setminus \{i,i'\} \\ |I| < k-2}} \prod_{j \in I} (e^{p_j v} - 1),$$

which is equivalent to

$$(e^{p_i v} - e^{p_{i'} v}) \sum_{\substack{I \subseteq [n] \setminus \{i,i'\} \\ |I| = k-2}} \prod_{j \in I} (e^{p_j v} - 1) = 0.$$

Hence, we have that $p_i = p_{i'}$ or that $|\{j : j \neq i, i', p_j > 0\}| < k-2$. To complete the proof, let us show that the minimum is not attained for any $\boldsymbol{p}$ such that, $|\mathsf{supp}(\boldsymbol{p})| < n$. We prove the latter using an induction on n. For clarity, we will use the notation f_n to indicate the relevant value of n and $\boldsymbol{p}_{\boldsymbol{u}}^n \triangleq (\frac{1}{n}, \ldots, \frac{1}{n})$. The base case in which $n = 2$ can be verified manually. This implies that $\boldsymbol{p} = \boldsymbol{p}_{\boldsymbol{u}}^2 = \left(\frac{1}{2}, \frac{1}{2}\right)$ is the only minimum point for f_2. Assume the claim holds up to n, and let us prove its correctness for $n+1$. Let $\boldsymbol{p} = (p_1, \ldots, p_{n+1})$ be a minimum point for f_{n+1}, and assume by contradiction that $|\mathsf{supp}(\boldsymbol{p})| < n+1$ and further assume w.l.o.g. that $p_{n+1} = 0$. Define $\boldsymbol{p}' = (p_1, \ldots, p_n)$ and note that $f_{n+1}(\boldsymbol{p}) = f_n(\boldsymbol{p}')$. By the induction assumption, we know that a minimum point of f_n has a support of size n and hence, by the analysis of the Lagrangian function, we have that f_n has a unique minimum point at $p_{\boldsymbol{u}}^n$. Therefore, we have that

$$f_{n+1}(\boldsymbol{p}) = f_n(\boldsymbol{p}') \geq f_n(\boldsymbol{p}_{\boldsymbol{u}}^n),$$

and equality is obtained if and only if $\boldsymbol{p}' = \boldsymbol{p}_{\boldsymbol{u}}^n$. Moreover, Claim 9.3 implies that $f_n(\boldsymbol{p}_{\boldsymbol{u}}^n) > f_{n+1}(\boldsymbol{p}_{\boldsymbol{u}}^{n+1})$, and thus,

$$f_{n+1}(\boldsymbol{p}) \geq f_n(\boldsymbol{p}_{\boldsymbol{u}}^n) > f_{n+1}(\boldsymbol{p}_{\boldsymbol{u}}^{n+1}),$$

which is a contradiction. Thus, we get that $|\mathsf{supp}(\boldsymbol{p})| = n+1$, which implies that the only minimum point of f_{n+1} is $\boldsymbol{p}_u^{n+1}$.

□

Theorem 9.5, together with the previous claims imply a lower bound on $\mathbb{E}[\nu_1^{\boldsymbol{p}}(n,k)]$, which is given next.

Corollary 9.6. *For any channel distribution $\boldsymbol{p}$ and any (n,k) code $\mathcal{C}$, it holds that,*

$$\mathbb{E}[\nu_1^{\boldsymbol{p}}(\mathcal{C})] \geq \mathbb{E}[\nu_1^{\boldsymbol{p}}(n,k)] \geq \mathbb{E}[\nu_1(n,k)] = \sum_{i=0}^{k-1} \frac{n}{n-i}, \tag{9.5}$$

and if $\lim_{n\to\infty} n - k = \infty$ then $\sum_{i=0}^{k-1} \frac{n}{n-i} \approx n \log(\frac{n}{n-k})$. Moreover (9.5) holds with equality if and only if $\boldsymbol{p} = \boldsymbol{p}_u$.

Finally, we give the asymptotic value for the minimum expected sample size for the noiseless channel, $\mathbb{E}[\nu_1(n,k)]$.

Theorem 9.7. *Let R be a constant, $0 < R < 1$. Then, we have that*

$$\lim_{n\to\infty} \frac{\mathbb{E}[\nu_1(n, k = \lfloor nR \rfloor)]}{k} = \frac{1}{R} \log\left(\frac{1}{1-R}\right).$$

Furthermore, consider a sequence of MDS codes $\{\mathcal{C}_i\}_{i=1}^{\infty}$ with parameters $[n_i, k_i]$ such that $\lim_{i\to\infty} k_i/n_i = 0$. Then,

$$\lim_{i\to\infty} \frac{\mathbb{E}[\nu_1(n_i, k_i)]}{k_i} = 1.$$

Proof. If $0 < R < 1$ is fixed, then n goes to infinity together with k and thus we have that,

$$\lim_{n\to\infty} \frac{\mathbb{E}[\nu_1(n, k = \lfloor nR \rfloor)]}{k} = \lim_{n\to\infty} \frac{n(H_n - H_{n-k})}{k} = \frac{1}{R} \log\left(\frac{1}{1-R}\right),$$

where the first equality holds from [14] and the second equality is a known result.

Furthermore, in the case in which we have a sequence of MDS codes $\{\mathcal{C}_i\}_{i=1}^{\infty}$, such that $\lim_{i\to\infty} \frac{k_i}{n_i} = 0$, the equality below holds.

$$\lim_{i\to\infty} \frac{\mathbb{E}[\nu_1(n_i, k_i)]}{k_i} = \lim_{i\to\infty} \frac{n_i(H_{n_i} - H_{n_i-k_i})}{k_i} = \lim_{i\to\infty} \frac{\sum_{j=0}^{k_i-1} \frac{n_i}{n_i-j}}{k_i},$$

where,

$$\lim_{i\to\infty} \frac{\sum_{j=0}^{k_i-1} \frac{n_i}{n_i-j}}{k_i} \leq \lim_{i\to\infty} \frac{k_i\left(\frac{n_i}{n_i-(k_i-1)}\right)}{k_i} = \lim_{i\to\infty} \frac{k_i\left(\frac{1}{1-\frac{k_i-1}{n_i}}\right)}{k_i} = 1,$$

and,

$$\lim_{i\to\infty} \frac{\sum_{j=0}^{k-1} \frac{n_i}{n_i-j}}{k_i} \geq \lim_{i\to\infty} \frac{k_i\left(\frac{n_i}{n_i-0}\right)}{k_i} = 1.$$

Thus, we can conclude that, $\lim_{i\to\infty} \frac{\mathbb{E}[\nu_1(n_i,k_i)]}{k_i} = 1$. □

9.5 The MDS Coverage Depth Problem - The Noisy Channel

The main goal of this section is to address Problem 9.1 for the noisy channel under the uniform distribution. Under this setup, we assume the data is encoded with an $[n,k]$ MDS code and that each strand $\boldsymbol{x}_i$ can be retrieved given some $t > 1$ reads, which are noisy copies of it, and cannot be retrieved given less than t reads. Similarly to the previous section, it is enough to successfully decode k (or more) sequences $\boldsymbol{x}_i$ in order to retrieve the stored information and so

under this setup the minimum sample size for our problem is equivalent to the quantity $\nu_t(n,k)$ where $t > 1$.

It should be noted that as listed in the related work section, the first part of Problem 9.1, i.e., the value of $\mathbb{E}[\nu_t(n,k)]$ is known [15] and is given in (9.1). However, it is not a closed-form expression, and in this section, we give several closed-form expressions that bound this value and thus extend the known result. Furthermore, the most related result regarding the probability distribution $P[\nu_t(n,k) > m]$ was given in [11]. The authors showed that for $n = k$ any $x \in \mathbb{R}$, the probability satisfies $P[\nu_t(n,n) > n\log n + (t-1)\log\log n + nx] \leq e^{-\frac{e^{-x}}{(t-1)!}}$.

In this section, we extend the latter result, by providing several bounds for the case when $k < n$, which is assumed for the rest of this section. Our main results for this case are stated in Theorem 9.8 and in Lemma 9.10. To discuss these results, we first define the following value. Given n, k, and t as stated above, we define

$$r(n,k,t) \triangleq n\log\left(\frac{n}{n-k}\right) + nt\log\log n + 2n\log(t+1). \tag{9.6}$$

In Theorem 9.8, it is shown that when n is large enough, the probability that more than $r(n,k,t)$ reads are required to retrieve the information i.e., $P[\nu_t(n,k) > r(n,k,t)]$ approaches zero.

Furthermore, for the case in which $k = Rn$, where $0 < R < 1$ is a fixed constant, the value of $r(n,k,t)$ can be reduced by replacing the expression $\log\log(n)$ with any function of n that approaches to infinity with n. To conclude this discussion, we also show in Lemma 9.10 that for any c, the probability that less than $n\log(\frac{n}{n-k}) - nc$ reads are enough to retrieve the information is bounded from above by $e^{-c}(1+\frac{1}{n-k})$. We start by showing that for any $\varepsilon > 0$, $P[\nu_t(n,k) \leq r(n,k,t)] \geq 1-\varepsilon$ for n large enough.

Theorem 9.8. *For any ε and n, such that $\varepsilon > 0$, $n > e^{\frac{6t\cdot 2^{t-1}}{\varepsilon}} \geq 16$, we have that,*

$$P[\nu_t(n,k) \leq r(n,k,t)] \geq 1-\varepsilon$$

Proof. To prove the statement in the theorem it is suffice to show that $P[\nu_t(n,k) > r(n,k,t)] < \varepsilon$. Denote $r \triangleq r(n,k,t)$ and recall that within the context of the urn problem (see Subsection 9.3.2), the random variable $\nu_t(n,k)$ denotes the number of balls (or rounds) until we have a set of k urns where each urn has at least t balls. Hence, we show that if the number of balls thrown is at least r, then the probability of having $n-k+1$ or more urns which are *not* filled with t balls is approaching zero. The approach leveraged in the proof is inspired by a technique first employed by Erdős and Rényi in [11]. Let us define the following event.

$E_t^{(r)}$**:** After r rounds, there exists a set S_t, of $n-k+1$ urns, each containing less than t balls.

Next, we show that the probability of $E_t^{(r)}$ approaches zero when n is large. To this end, we define $z_i(n,r)$ for $1 \leq i \leq n$, as a random variable that governs the number of balls in the i-th urn, after r draws. For n large enough, the probability that urn i has at most $t-1$ balls after r

draws is denoted by $P[z_i(n,r) \leq t-1]$ and is given by,

$$\begin{aligned} P[z_i(n,r) \leq t-1] &= \sum_{j=0}^{t-1} \binom{r}{j} \left(\frac{1}{n}\right)^j \left(1-\frac{1}{n}\right)^{r-j} \\ &\leq t \cdot \binom{r}{t-1} \left(\frac{1}{n}\right)^{t-1} \left(1-\frac{1}{n}\right)^{r-(t-1)} \\ &\leq t \cdot \left(\frac{r \cdot e}{t-1}\right)^{t-1} \left(\frac{1}{n}\right)^{t-1} \left(1-\frac{1}{n}\right)^{r-(t-1)}, \end{aligned}$$

where the first inequality is proven in Claim 9.35 in Appendix A, and the last inequality follows from the fact that $\binom{r}{t-1} \leq (\frac{re}{t-1})^{t-1}$. Note that $(\frac{e}{t-1})^{t-1} < 3$, for $t > 1$. Thus,

$$P[z_i(n,r) \leq t-1] \leq 3t \cdot \left(\frac{r}{n}\right)^{t-1} \left(1-\frac{1}{n}\right)^{n\left(\frac{r}{n}-\frac{t-1}{n}\right)}.$$

We have that,

$$\begin{aligned} P[z_i(n,r) \leq t-1] &\leq 3t \cdot \left(\log\left(\frac{n}{n-k}\right) + t\log\log(n) + 2\log(t+1)\right)^{t-1} \left(e^{\left(\frac{-r}{n}+\frac{t-1}{n}\right)}\right) \\ &\leq 3t \cdot (2\log n)^{t-1} \left(\frac{n-k}{n}\right) \left(\frac{1}{\log^t n}\right) \left(\frac{1}{(t+1)^2}\right) e^{\frac{t-1}{n}} \\ &= 3t \cdot \frac{e^{\frac{t-1}{n}}}{(t+1)^2} \cdot \frac{(2\log n)^{t-1}}{\log^t(n)} \cdot \frac{n-k}{n} \\ &= 3t \cdot \frac{e^{\frac{t-1}{n}}}{(t+1)^2} \cdot \frac{2^{t-1}}{\log(n)} \cdot \frac{n-k}{n}, \end{aligned}$$

where the second inequality holds since for n large enough $\log(\frac{n}{n-k}) + t\log\log(n) + 2\log(t+1) \leq (2\log n)$. It should be noted that for $n > t$, which is the case of our interests, we have that $3t \cdot \frac{e^{\frac{t-1}{n}}}{(t+1)^2} \leq 6t$, and hence,

$$P[z_i(n,r) \leq t-1] \leq 6t \cdot \frac{2^{t-1}}{\log(n)} \cdot \frac{n-k}{n}.$$

Now let us define a random variable Y as the number of urns with less than t balls. From the linearity of expectation, regardless if the urns are independent or not, the expected number of urns that have less than t balls is,

$$\begin{aligned} \mathbb{E}[Y] &= \Sigma_{i=1}^{n} \mathbb{E}[z_i(n,r)] \\ &= n \cdot P[z_i(n,r) \leq t-1] \leq (n-k) \cdot 6t \cdot \frac{2^{t-1}}{\log(n)}, \end{aligned}$$

where the last inequality holds for n large enough.

Note that

$$P\left[E_t^{(r)}\right] = P[Y \geq n-k+1],$$

and hence by Markov's inequality, we can conclude that,

$$P[Y \geq n-k+1] \leq \frac{\mathbb{E}[Y]}{n-k+1} < 6t \cdot \frac{2^{t-1}}{\log(n)}.$$

Thus, we get that $P[E_t^{(r)}] \to 0$ for n large enough which implies the statement in the theorem. □

For fixed-rate codes, i.e., for the case where $k = Rn$, when $0 < R < 1$, and R is a fixed constant (when n grows), we present a stronger result in the next theorem. The proof of this theorem can be found in Appendix A.

Theorem 9.9. *Let $f : \mathbb{N} \to \mathbb{R}$ be a function such that $\lim_{n\to\infty} f(n) = \infty$, and let*

$$r_f(n, k = Rn, t) \triangleq n \log\left(\frac{1}{1-R}\right) + ntf(n) + 2n(t+1). \tag{9.7}$$

Then, for n large enough, it holds that

$$P\left[\nu_t(n,k) > r_f(n,k,t)\right] \leq 6t^t \frac{(2 \cdot f(n))^{t-1}}{e^{t \cdot f(n)}} \cdot (1-R).$$

Theorem 9.9 draws a connection between the sample size and the probability of successful retrieval when using fixed-rate codes. In particular, using the results of Theorem 9.9 one can pick any function $f(n)$ that approaches infinity as slowly (or fast) as possible to get an upper bound on this probability which gets bigger (or smaller).

Next, for any $c \in \mathbb{R}$, we denote,

$$r_L(n,k,c) \triangleq n \log\left(\frac{n}{n-k}\right) - nc. \tag{9.8}$$

In the next lemma, an upper bound on the probability $P[\nu_t(n,k) \leq r_L(n,k,c)]$ is given.

Lemma 9.10. *For any $c > 0$, and any $t \geq 1$ it holds that,*

$$P\left[\nu_t(n,k) \leq n \log(\frac{n}{n-k}) - nc\right] \leq e^{-c}\left(1 + \frac{1}{n-k}\right).$$

Proof. We first highlight that $\nu_t(n,k) \geq \nu_1(n,k)$, and thus it is enough to show that

$$P\left[\nu_1(n,k) \leq n \log(\frac{n}{n-k}) - nc\right] \leq e^{-c}\left(1 + \frac{1}{n-k}\right).$$

We have that,

$$
\begin{aligned}
e^{\log(\frac{n}{n-k})-c} \cdot \mathbb{E}\left[e^{-\frac{\nu_1(n,k)}{n}}\right] &= e^{\log(\frac{n}{n-k})-c} \cdot \sum_{j=1}^{\infty} e^{-\frac{j}{n}} P[\nu_1(n,k)=j] \\
&= \sum_{j=1}^{\infty} e^{\log(\frac{n}{n-k})-c-\frac{j}{n}} P[\nu_1(n,k)=j] \\
&= \sum_{j=1}^{\lfloor n\log(\frac{n}{n-k})-nc\rfloor} e^{\log(\frac{n}{n-k})-c-\frac{j}{n}} P[\nu_1(n,k)=j] \\
&\quad + \sum_{j=1+\lfloor n\log(\frac{n}{n-k})-nc\rfloor}^{\infty} e^{\log(\frac{n}{n-k})-c-\frac{j}{n}} P[\nu_1(n,k)=j] \\
&\geq \sum_{j=1}^{\lfloor n\log(\frac{n}{n-k})-nc\rfloor} e^{\log(\frac{n}{n-k})-c-\frac{j}{n}} P[\nu_1(n,k)=j] \\
&\geq \sum_{j=1}^{\lfloor n\log(\frac{n}{n-k})-nc\rfloor} 1 \cdot P[\nu_1(n,k)=j] \\
&\geq P\left[\nu_1(n,k) \leq n\log\left[\frac{n}{n-k}\right]-nc\right].
\end{aligned}
$$

From [25], the generating function of the geometric random variable $\nu_1(n,k)$ is given by

$$
G_{\nu_1(n,k)}(x) = \mathbb{E}[x^{\nu_1(n,k)}] = \sum_{j=0}^{\infty} P[\nu_1(n,k)=j]x^j = \prod_{i=1}^{k} \frac{(n-(i-1))x}{n-(i-1)x} = \prod_{i=1}^{k} \frac{(1-\frac{i-1}{n})x}{1-\frac{i-1}{n}x}.
$$

Thus, given $x = e^{-1/n}$, we get that,

$$
\begin{aligned}
\mathbb{E}\left[e^{\frac{-\nu_1(n,k)}{n}}\right] &= \prod_{i=1}^{k} \frac{(1-\frac{i-1}{n})e^{-\frac{1}{n}}}{1-(\frac{i-1}{n})e^{-\frac{1}{n}}} \\
&= \prod_{i=1}^{k} \frac{1-\frac{i-1}{n}}{e^{\frac{1}{n}}-\frac{i-1}{n}} \leq \prod_{i=1}^{k} \frac{1-\frac{i-1}{n}}{1+\frac{1}{n}-\frac{i-1}{n}} \\
&= \prod_{i=1}^{k} \frac{1-\frac{i-1}{n}}{1-\frac{i-2}{n}} = \frac{1-\frac{k-1}{n}}{1+\frac{1}{n}} = \frac{n-k+1}{n+1} \leq \frac{n-k+1}{n},
\end{aligned}
$$

where in the first inequality we used the fact the $e^{\frac{1}{n}} \geq 1+\frac{1}{n}$. Hence, it holds that, for positive c,

$$
e^{\log(\frac{n}{n-k})-c} \cdot \mathbb{E}\left[e^{-\nu_1(n,k)/n}\right] \leq e^{-c} \cdot \frac{n}{n-k} \cdot \frac{n-k+1}{n} = e^{-c}\left(1+\frac{1}{n-k}\right).
$$

Finally, we conclude that,

$$
P\left[\nu_1(n,k) \leq n\log(\frac{n}{n-k})-nc\right] \leq e^{\log(\frac{n}{n-k})-c} \cdot \mathbb{E}\left[e^{-\nu_1(n,k)/n}\right] \leq e^{-c}\left(1+\frac{1}{n-k}\right).
$$

□

Combining Lemma 9.10 and Theorem 9.8, and assuming t is a constant with respect to n, in the next theorem we show upper and lower bounds on $\mathbb{E}\left[\frac{\nu_t(n,k)}{n}\right]$.

Theorem 9.11. *For any $\varepsilon > 0$, there exists n_ε, such that for any $n > n_\varepsilon$ we have that,*

$$\log\left(\frac{1}{1-R}\right) + f_c(n,R) \leq \mathbb{E}\left[\frac{\nu_t(n,k)}{n}\right] \leq \left(\log\left(\frac{1}{1-R}\right) + t\log\log n + 2\log(t+1)\right)\cdot(1+2\varepsilon),$$

where $f_c(n,R) = \frac{1}{2n}(1-\frac{1}{1-R}) - \sum_{h=1}^{\infty}\frac{B_{2h}}{2hn^{2h}}\left(1-\frac{1}{(1-R)^{2h}}\right) = \mathcal{O}(\frac{1}{n^2})$, and B_h denotes the h-th Bernoulli number.

Proof. First, we highlight that for any integer $t > 0$, it holds that $\nu_t(n,k) \geq \nu_1(n,k)$. Next, we recall the known results proven in [15], where they showed $\mathbb{E}[\nu_1(n,k)] = n(H_n - H_{n-k})$. Hence, we can conclude that the following holds for n large enough,

$$\begin{aligned}
\mathbb{E}[\nu_t(n,k)] &\geq \mathbb{E}[\nu_1(n,k)] \\
&= n(H_n - H_{n-k}) \\
&= n\left(\log(n) + \gamma + \frac{1}{2n} - \sum_{h=1}^{\infty}\frac{B_{2h}}{2hn^{2h}} - \log(n-k) - \gamma - \frac{1}{2(n-k)} + \sum_{h=1}^{\infty}\frac{B_{2h}}{2h(n-k)^{2h}}\right) \\
&= n\log\left(\frac{n}{n-k}\right) + \frac{1}{2}\left(1-\frac{1}{1-R}\right) - n\sum_{h=1}^{\infty}\frac{B_{2h}}{2hn^{2h}}\left(1-\frac{1}{(1-R)^{2h}}\right) \\
&= n\log\left(\frac{1}{1-R}\right) + nf_c(n,R),
\end{aligned}$$

where $\gamma \sim 0.5772156649$ is the Euler-Mascheroni constant, where the last equality was proven in [15]. Next, let $r_n \triangleq r(n,k,t)$ (recall that by (9.6), $r(n,k,t) = n\log(\frac{1}{1-R}) + nt\log\log(n) + 2n\log(t+1)$). In Theorem 9.8 we showed that $P[\nu_t(n,k) > r_n] < \varepsilon$. Using the same methods, in Appendix B we proved Theorem 9.36 which states that for any integer $i \geq 1$ and for n large enough, $P[\nu_t(n,k) > r_n \cdot i] < \varepsilon \cdot \frac{i^{t-1}}{\log^{t(i-1)}(n)}$, and thus we can conclude

that,

$$\begin{aligned}
E[\nu_t(n,k)] &= \sum_{r\in\mathbb{N}} P(\nu_t(n,k)\geq r) \\
&= \sum_{r<r_n} P(\nu_t(n,k)\geq r) + \sum_{r\geq r_n} P(\nu_t(n,k)\geq r) \\
&\leq 1\cdot r_n + \sum_{r\geq r_n} P(\nu_t(n,k)\geq r) \\
&= r_n + \sum_{i=1}^{\infty} \sum_{r=i\cdot r_n}^{(i+1)\cdot r_n} P(\nu_t(n,k)\geq r) \\
&\leq r_n + \sum_{i=1}^{\infty} \sum_{r=i\cdot r_n}^{(i+1)\cdot r_n} P(\nu_t(n,k)\geq i\cdot r_n) \\
&= r_n + \sum_{i=1}^{\infty} r_n\cdot P(\nu_t(n,k)\geq i\cdot r_n) \\
&< r_n + \sum_{i=1}^{\infty} r_n\cdot\varepsilon\cdot\frac{i^{t-1}}{\log^{t(i-1)}(n)} \\
&= r_n + \varepsilon\cdot r_n \sum_{i=1}^{\infty}\frac{i^{t-1}}{\log^{t(i-1)}(n)} \\
&\overset{(a)}{<} r_n + 2\varepsilon\cdot r_n \\
&= r_n\cdot(1+2\varepsilon),
\end{aligned}$$

where (a) follows since $\sum_{i=1}^{\infty}\frac{i^{t-1}}{\log^{t(i-1)}(n)} < 2$ for n large enough and any integer $t>0$. Lastly, we simplify the expression,

$$\begin{aligned}
\frac{1}{n}r_n(1+2\varepsilon) &= \left(\log\left(\frac{1}{1-R}\right) + t\log\log n + 2\log(t+1)\right)\cdot(1+2\varepsilon) \\
&= \log\left(\frac{1}{1-R}\right) + \mathcal{O}(t\log\log n),
\end{aligned}$$

which completes the proof. □

For practical purposes of DNA storage systems, it is sometimes required to plan ahead and sample the number of reads that guarantees successful decoding with high probability. Hence, we turn to the following strongly related problem and give a closed-form expression to the corresponding value. Turning back to the urn problem terminology, we define $X^{(r)}$ as the number of urns that are not filled with at least t balls after r rounds. The goal is to find a lower bound on the number of rounds r, that guarantees that the expected number of urns that are *not filled* with t balls is at most $n-k$. That is, to find r_E, such that for any $r\geq r_E$, we have that $\mathbb{E}[X^{(r)}]\leq n-k$. In order to derive this result, we first consider the probability that any fixed

urn is *not filled* with t or more balls by the r-th round. This probability is given by,

$$p = \sum_{j=0}^{t-1} \binom{r}{j} n^{-j} \left(1 - \frac{1}{n}\right)^{r-j} \leq e^{-rD(\frac{t-1}{r}||\frac{1}{n})},$$

where the last inequality follows from Chernoff bound [8] for $r \geq n(t-1)$, and $D(a||p)$ is the Kullback–Leibler divergence [9] which is given by

$$D(a||p) \triangleq a \log_2 \frac{a}{p} + (1-a) \log_2 \frac{1-a}{1-p}.$$

Under our setup, each of the n urns can be interpreted as a Bernoulli random variable with probability p, which is denoted by $X_i^{(r)}$ for $1 \leq i \leq n$. Note that $X^{(r)} = \sum_{i=1}^{n} X_i^{(r)}$ is the number of urns that are not filled with at least t balls after r rounds, which implies that the number of urns that have at least t balls is $n - X^{(r)}$. Our approach will be to determine a value for r, which guarantees (in expectation) that $X^{(r)}$ is at most $n-k$. From the linearity of expectation,

$$\mathbb{E}[X^{(r)}] = np \leq ne^{-(t-1)\log_2(\frac{n(t-1)}{r}) - (r-(t-1))\log_2\left(\frac{(r-(t-1))n}{r(n-1)}\right)}. \tag{9.9}$$

The next claim will be used in the derivation to follow and its proof can be found in Appendix C.

Claim 9.12. *For $r \geq n(t-1)$, we have that $\mathbb{E}[X^{(r)}] \leq n-k$, if,*

$$-\frac{r}{n(t-1)} e^{-\frac{r}{n(t-1)}} \geq -\frac{1}{e}\left(1 - \frac{k}{n}\right)^{\frac{\log 2}{t-1}}. \tag{9.10}$$

Using known results on the Lambert W function [10, Section IV], [7, Theorem 1], the values of r for which (9.10) holds can be concluded. This is summarized in the next theorem, and the complete proof can be found in Appendix C. For $0 < R < 1$, we denote,

$$r_E(n, k = Rn, t) \triangleq n(t-1) - n\log 2 \log(1-R) + n(t-1)\sqrt{-\frac{2\log 2}{t-1}\log(1-R)}. \tag{9.11}$$

Theorem 9.13. *Let $R = \frac{k}{n}$. For any $r \geq r_E(n,k,t)$, we have that $\mathbb{E}[X^{(r)}] \leq n-k$.*

At this point, we would like to shed some light on the relation between Theorem 9.8, Theorem 9.9 and Theorem 9.13. In our setup, which uses the urn problem terminology, it is assumed that r balls are thrown into n unique urns, and we are interested in the event that at least k of these urns contain at least t balls each. This scenario can be parameterized in two different ways; (a) the number of balls that need to be thrown, and (b) the number of urns that contain $t-1$ or less balls. The random variable $\nu_t(n,k)$ governs the value in (a), assuming that the value in (b) is fixed. Analogously, the random variable $X^{(r)}$ governs the value in (b), assuming that the value in (a) is fixed.

In the case where the probability distribution is tightly concentrated (i.e., where $\nu_t(n,k)$ is tightly concentrated around its mean and similarly for $X^{(r)}$), one would expect these two quantities to coincide. Figure 9.2, shows results from computer simulations we made to demonstrate the results of Theorem 9.8 and Theorem 9.13. In the presented simulation we used $n = 100,000$ urns, $R \in \{0.5, 0.8\}$, $k = Rn$, and $t = 5$. In each simulation r balls are drawn, each inserted into one of the urns randomly, and the simulation is considered as success if it ends with at least k urns, each with at least t balls. For any value of r, the presented result is the fraction of successful simulations out of $1,000$ simulations we have made per r. The Y-axis shows the fraction of successful experiments, and the X-axis shows the number of draws r normalized by $n\log(\frac{1}{1-R})$. It can be seen that the success rate of both values presented in Theorem 9.8 ($r(n,k,t)$) and Theorem 9.13 ($r_E(n,k,t)$) are 1.

Practically speaking, as mentioned above, the noisy channel fits the real scenario of DNA storage systems. Hence, it should be mentioned that a similar problem was studied experimentally by Erlich and Zielinski [12], however, with a slightly different setup. They presented the DNA fountain, a Luby transform-based scheme and assumed that the total number of reads is fixed and is given (from the DNA sequencer) and it is distributed with a negative binomial distribution. Thus, they were able to calculate the average number of copies per strand and empirically evaluate the required sample size as a function of the distribution's parameters. It should be noted that they only considered reads of the design length and thus the error rates were reduced. They also evaluated how dilution affects the distribution and the required sample size.

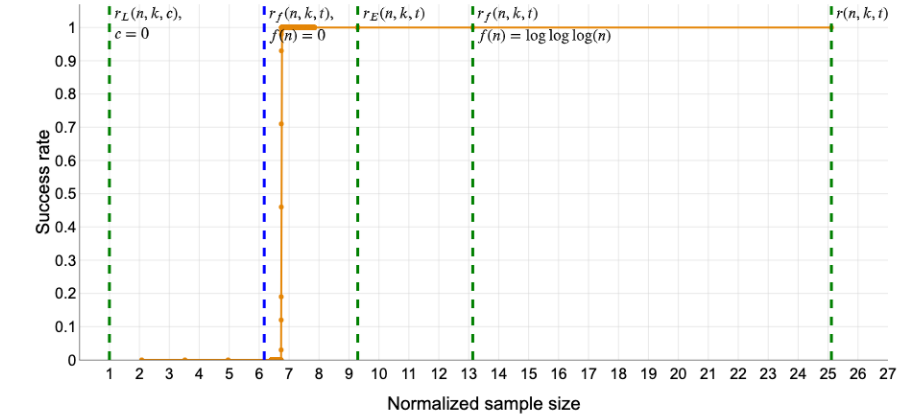
Finally, another variation of the noisy channel $\mathcal{S}$ is studied, which is relevant to the DNA fountain [12] and similar schemes. Here, it is required to obtain a single noiseless copy from k out of the n synthesized strands. Assuming uniform distribution on the strands, in this channel, any sampled read is drawn noiseless with some fixed probability $0 < \alpha < 1$. We use the notation of $\omega_\alpha(n,k)$ to denote the random variable describing the required sample size to ensure successful decoding in this case. We note that this setup is easier to analyze, and the following results can be derived using similar techniques as in the classical coupons collector's problem [14]; see Appendix D.

Theorem 9.14. *For any $k \leq n$, $\mathbb{E}[\omega_\alpha(n,k)] = \frac{n}{\alpha}(H_n - H_{n-k})$.*

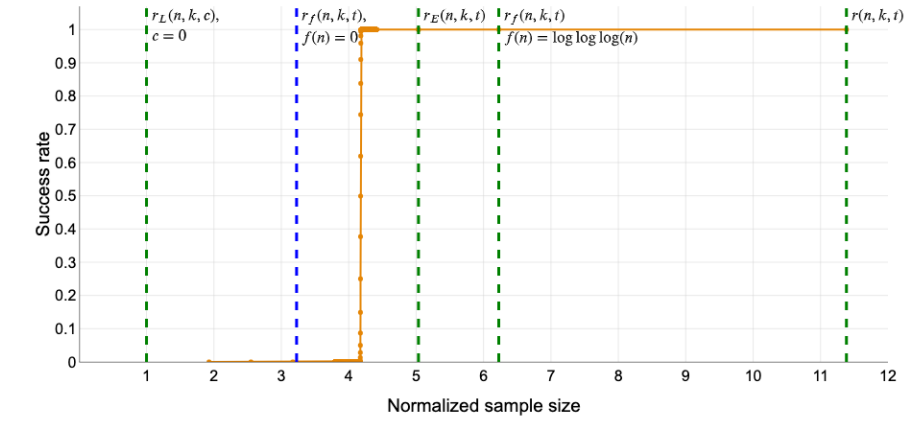
9.6 Random Access

In this section we study the problem of optimizing the sample size for random access queries in DNA storage systems. Recall that, in this problem, a vector of k information strands each of length ℓ, $\mathbf{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k) \in (\Sigma^\ell)^k$, is encoded into a vector of n strands, each of length ℓ, $\mathbf{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) \in (\Sigma^\ell)^n$ that are stored in the DNA storage channel as described in Section 9.2. Later, the user wishes to retrieve a single information strand $\boldsymbol{u}_i$ for some $i \in [k]$. Unless stated otherwise, we assume the channel is uniform and noiseless.

We start by studying the case when the number of information strands matches the number of coded strands (i.e. $n = k$), and prove that the optimal retrieval strategy involves no coding, resulting in an expected retrieval time of k. Next, we extend our insights to more involved cases, including systematic MDS codes, affirming that the expected retrieval time remains k for any



$R = 0.5$



$R = 0.8$

Figure 9.2: Simulation results of the success rate (fraction of successful experiments) as a function of the number of draws. The X-axis shows the number of draws (normalized by $n\log(\frac{1}{1-R})$), while the Y-axis shows the fraction of simulations in which there were at least k urns with t balls each. The parameters used in the simulations were $n = 100,000$, $t = 5$, and for each number of draws, we had $1,000$ simulations. It can be seen that for both Theorem 9.8 and Theorem 9.13 the success rate of 1, as expected.

information strand. Finally, we present explicit code constructions achieving expected retrieval times below k and evaluate their performance analytically and through simulations, while also providing lower bounds on the maximum expected retrieval time in different scenarios.

9.6.1 Preliminary Results

Recall that given an (n,k) code $\mathcal{C}$, for $i \in [k]$, we denote by $\tau_i(\mathcal{C})$ the random variable that governs the number of samples to recover the i-th information strand. The next lemma fully solves Problem 9.3 when no coding is used.

Lemma 9.15. *Let $n \geq 1$. For any $1 \leq i \leq n$, we have that*

1) $\mathbb{E}[\tau_i] = n$ and $T_{\max} = T_{avg} = n$.

2) For any $r \in \mathbb{N}$ we have that $P[\tau_i > r] = \left(1 - \frac{1}{n}\right)^r$, and $P[\tau_i = r] = \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{r-1}$.

Proof. For the first part, note that for any i, τ_i has geometric distribution with success probability $p = \frac{1}{n}$ and hence we have that $\mathbb{E}[\tau_i] = p^{-1} = n$ which implies that

$$T_{\max} = \max_{1 \leq i \leq n} \mathbb{E}[\tau_i] = p^{-1} = n,$$

and

$$T_{\text{avg}} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}[\tau_i] = \frac{np^{-1}}{n} = p^{-1} = n.$$

For the second part we have that $\tau_i > r$ for an integer r only if $\boldsymbol{u}_i$ was not sampled in the first r trials, and hence

$$P[\tau_i > r] = (1-p)^r = \left(1 - \frac{1}{n}\right)^r,$$

and

$$P[\tau_i = r] = \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{r-1}.$$

□

Before we continue with the analysis of more involved cases, we define the n random variables $\widehat{\tau}_i(\mathcal{C}), i \in [n]$, such that $\widehat{\tau}_i(\mathcal{C})$ governs the required sample size to retrieve the i-th encoded strand. Additionally, for every set $J \subseteq [n]$, let $\widehat{\tau}_J(\mathcal{C}) \triangleq \max_{i \in J} \widehat{\tau}_i(\mathcal{C})$. These random variables are used as a technical tool in our analysis and the key idea is given in the next lemma. The proof follows the same ideas as the proof for the coupon collector's problem and is given here for completeness.

Claim 9.16. *For any (n,k) code $\mathcal{C}$ and any $J \subseteq [n]$ of size ρ we have that $\mathbb{E}[\widehat{\tau}_J(\mathcal{C})] = nH_\rho$.*

Proof. Let t_i for $1 \leq i \leq \rho$ be the number of draws to collect the i-th strand in J after the $(i-1)$-th strand from J was collected. Note that $\widehat{\tau}_J(\mathcal{C}) = \sum_{i=1}^{\rho} t_i$. Additionally, observe that t_i is a geometric random variable with success probability $p_i = \frac{\rho - i + 1}{n}$ and $\mathbb{E}[t_i] = \frac{1}{p_i}$. Hence, by the linearity of the expectation we have that

$$\mathbb{E}[\widehat{\tau}_J(\mathcal{C})] = \mathbb{E}\left[\sum_{i=1}^{\rho} t_i\right] = \sum_{i=1}^{\rho} \mathbb{E}[t_i] = \sum_{i=1}^{\rho} \frac{n}{\rho - i + 1} = n \sum_{i=1}^{\rho} \frac{1}{i} = nH_\rho.$$

□

For the rest of this section, it is assumed that $\mathcal{C}$ is an (n,k) code and $\mathbf{X}$ is the encoded codeword of the information vector $\mathbf{U}$. The structure of $\mathcal{C}$ defines for each information strand all the possible sets of encoded strands that are sufficient for its recovery. This concept is similar to recovery sets in *locally repairable codes* [26] as well as the ones with *availability* [13, 19, 31].

This can be defined formally as follows.

Definition 9.17. *Let $\mathcal{C}$ be an (n,k) code. We say that $J \subseteq [n]$ is a* retrieval set *of the i-th information strand (i.e., $\boldsymbol{u}_i$) if it is possible to decode the information strand $\boldsymbol{u}_i$ from the encoded strands whose indices belong to J. The set of all retrieval sets of $\boldsymbol{u}_i$ is denoted by $\widehat{\mathcal{D}}(i)$, and $\mathcal{D}(i)$ is the set of all minimal retrieval sets of $\boldsymbol{u}_i$ (with respect to the inclusion relation).*

We say that an (n,k) code $\mathcal{C}$ is a *systematic* code if for any $i \in [k]$ it holds that $\boldsymbol{u}_i$ has a retrieval set of size one. In other words, $\mathcal{C}$ is systematic if for any $i \in [k]$ we have that

$$\min\{|J| : J \in \mathcal{D}(i)\} = 1.$$

Next, we consider the case of non-systematic codes for $k = n$ (in particular $\mathbf{U} \neq \mathbf{X}$). Since $\mathbf{X}$ and $\mathbf{U}$ have the same length, given any set of strands $\{\boldsymbol{x}_i : i \in J\}$, we can recover at most $|J|$ information strands from $\mathbf{U}$. Our goal is to extend Lemma 9.15 to the coded case when $k = n$ using this basic insight.

Claim 9.18. *For any code $(n = k, k)$ $\mathcal{C}$, we have that $T^{\mathcal{C}}_{\max} \geq T_{\max} = n$ and $T^{\mathcal{C}}_{avg} \geq T_{avg} = n$, where equality is obtained if and only if $\mathcal{C}$ is systematic. In particular, if we let ρ_i be the size of the smallest retrieval set for the information strand $\boldsymbol{u}_i$, then*

1) $\mathbb{E}[\tau_i(\mathcal{C})] = nH_{\rho_i}$,
2) $T^{\mathcal{C}}_{\max} = nH_\rho$, *where* $\rho \triangleq \max_i \rho_i$,
3) $T^{\mathcal{C}}_{avg} = \sum_{i=1}^{n} H_{\rho_i}$.

Proof. If each $\boldsymbol{u}_i$ can be retrieved from a single strand $\boldsymbol{x}_j$ (i.e., $\mathcal{C}$ is a systematic code), then similarly to the proof of Lemma 9.15 we have that $T^{\mathcal{C}}_{\max} = T^{\mathcal{C}}_{\text{avg}} = \mathbb{E}[\tau_i(\mathcal{C})] = n$, for any $i \in [n]$. Otherwise, assume w.l.o.g. that $\boldsymbol{u}_1$ cannot be retrieved from a single strand and let $J \subseteq [n]$ be a set of minimal size $|J| = \rho_1$ such that $J \in \mathcal{D}(1)$. By the latter observation and

since it is possible to retrieve any information strand $\boldsymbol{u}_i$ from all the n strands, the fact that $n = k$ implies that if there exists $J' \subseteq [n]$, such that J' is a retrieval set of $\boldsymbol{u}_1$ (i.e., $J' \in \widehat{\mathcal{D}}(1)$) then J' contains J. Hence, $|\mathcal{D}(1)| = 1$, i.e., the set J is the only minimal retrieval set of $\boldsymbol{u}_1$, and by Claim 9.16, we have that $\mathbb{E}[\tau_i(\mathcal{C})] = \mathbb{E}[\widehat{\tau}_J(\mathcal{C})] = nH_{\rho_1} > n$, where the last inequality holds since $|J| = \rho_1 > 1$. Thus,

$$T^{\mathcal{C}}_{\max} = \max_{1 \le i \le k} \mathbb{E}[\tau_i(\mathcal{C})] = \max_{1 \le i \le n} nH_{\rho_i} = nH_{\rho},$$

and

$$T^{\mathcal{C}}_{\text{avg}} = \frac{1}{k} \sum_{i=1}^{k} \mathbb{E}[\tau_i(\mathcal{C})] = \frac{1}{n} \sum_{i=1}^{n} nH_{\rho_i} = \sum_{i=1}^{n} H_{\rho_i}.$$

Note that $H_{\rho_i} \ge 1$ for any $i \in [k]$ and since $\rho_1 > 1$, we have that $H_{\rho_1} > 1$. Hence $T^{\mathcal{C}}_{\max} > n$ and $T^{\mathcal{C}}_{\text{avg}} > n$ which completes the proof. □

9.6.2 The Singleton Coverage Depth Problem

We continue by studying cases where $n > k$. Next, the case where the minimal retrieval sets are disjoint is considered. We start with the case in which a strand $\boldsymbol{x}_i$ has exactly two minimal retrieval sets $\mathcal{D}(i) = \{A, B\}$ and $A \cap B = \emptyset$, while the next example considers the simple parity code which is a special instance of this case.

Example 9.19. *Let $\mathcal{C}$ be the $(4, 3)$ parity code. We have that $\mathbf{X} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3, \boldsymbol{x}_4)$, where*

$$\boldsymbol{x}_4 = \boldsymbol{u}_1 + \boldsymbol{u}_2 + \boldsymbol{u}_3.$$

Since the code is symmetric, let us consider w.l.o.g. $\boldsymbol{u}_1$. Note that $\mathcal{D}(1) = \{\{\boldsymbol{u}_1\}, \{\boldsymbol{u}_2, \boldsymbol{u}_3, \boldsymbol{x}_4\}\}$ and the two retrieval sets are disjoint. Hence, we cannot recover $\boldsymbol{u}_1$ from a series of r draws only if the series of draws does not contain $\boldsymbol{u}_1$, and it either contains one unique strand or two unique strands. Hence,

$$\begin{aligned}\mathbb{E}[\tau_1(\mathcal{C})] &= \sum_{r=0}^{\infty} P[\tau_i(\mathcal{C}) > r] = 1 + \sum_{r=1}^{\infty} \left(3 \cdot \frac{1}{4^r} + \binom{3}{2} \sum_{j=1}^{r-1} \binom{r}{j} \frac{1}{4^j} \cdot \frac{1}{4^{r-j}} \right) \\ &= 1 + 3 \sum_{r=1}^{\infty} \frac{1}{4^r} + 3 \sum_{r=1}^{\infty} \frac{2^r - 2}{4^r} = 1 + 3 \cdot \frac{1}{3} + 3 \cdot \frac{1}{3} = 3.\end{aligned}$$

That is, in this case, $\mathbb{E}[\tau_1(\mathcal{C})] = k$.

The next theorem extends Example 9.19 to any code $\mathcal{C}$ and an information strand $\boldsymbol{x}_i$ with exactly two minimal retrieval sets A, B such that $A \cap B = \emptyset$.

Theorem 9.20. *Let $\mathcal{C}$ be an (n, k) code and $i \in [k]$. If $\mathcal{D}(i) = \{A, B\}$, for two disjoint retrieval sets, $A \cap B = \emptyset$, then $\mathbb{E}[\tau_i(\mathcal{C})] = n \cdot \left(H_{|A|} + H_{|B|} - H_{|A|+|B|} \right)$.*

Proof. Denote $\rho_A = |A|, \rho_B = |B|$. For a set of indices $J \subseteq [n]$, let $\lambda_J(r-1)$ be the number of different options to draw strands in the first $r-1$ draws such that for at least one of the indices $j \in J$, the strand $\boldsymbol{x}_j$ was not drawn. Additionally, let $\lambda(r-1)$ be the number of different options to draw strands in the first $r-1$ draws such that the i-th information strand cannot be retrieved from the set of drawn strands. Note that since $\mathcal{D}(i) = \{A, B\}$, we have that $\lambda(r-1)$ is the number of different options to draw strands in the first $r-1$ draws such that at least one strand from A and at least one strand from B were not drawn. Hence.

$$\lambda_{A\cup B}(r-1) = \lambda_A(r-1) + \lambda_B(r-1) - \lambda(r-1),$$

and

$$\begin{aligned}
\lambda(r-1) &= \lambda_A(r-1) + \lambda_B(r-1) - \lambda_{A\cup B}(r-1) \\
&\overset{(a)}{=} \sum_{j=1}^{\rho_A} \binom{\rho_A}{j} (-1)^{j+1} (n-j)^{r-1} \\
&\quad + \sum_{j=1}^{\rho_B} \binom{\rho_B}{j} (-1)^{j+1} (n-j)^{r-1} \\
&\quad - \sum_{j=1}^{\rho_A+\rho_B} \binom{\rho_A+\rho_B}{j} (-1)^{j+1} (n-j)^{r-1},
\end{aligned}$$

where (a) follows from the inclusion-exclusion principle. Using the tail sum formula for the expectation, we have that

$$\begin{aligned}
\mathbb{E}[\tau_i(\mathcal{C})] &= \sum_{r=1}^{\infty} \frac{\lambda(r-1)}{n^{r-1}} = \sum_{r=1}^{\infty} \sum_{j=1}^{\rho_A} \frac{\binom{\rho_A}{j}(-1)^{j+1}(n-j)^{r-1}}{n^{r-1}} \\
&\quad + \sum_{r=1}^{\infty} \sum_{j=1}^{\rho_B} \frac{\binom{\rho_B}{j}(-1)^{j+1}(n-j)^{r-1}}{n^{r-1}} \\
&\quad - \sum_{r=1}^{\infty} \sum_{j=1}^{\rho_A+\rho_B} \frac{\binom{\rho_A+\rho_B}{j}(-1)^{j+1}(n-j)^{r-1}}{n^{r-1}}.
\end{aligned}$$

Next, we analyze the first term in the latter expression and the other two terms can be

analyzed similarly.

$$\begin{aligned}
&\sum_{r=1}^{\infty}\sum_{j=1}^{\rho_A}\frac{\binom{\rho_A}{j}(-1)^{j+1}(n-j)^{r-1}}{n^{r-1}}\\
=&\sum_{r=1}^{\infty}\sum_{j=1}^{\rho_A}\binom{\rho_A}{j}(-1)^{j+1}\left(1-\frac{j}{n}\right)^{r-1}\\
\overset{(a)}{=}&\sum_{j=1}^{\rho_A}\binom{\rho_A}{j}(-1)^{j+1}\sum_{r=1}^{\infty}\left(1-\frac{j}{n}\right)^{r-1}\\
\overset{(b)}{=}&\sum_{j=1}^{\rho_A}\binom{\rho_A}{j}(-1)^{j+1}\frac{n}{j}=n\sum_{j=1}^{\rho_A}\binom{\rho_A}{j}\frac{(-1)^{j+1}}{j}\\
\overset{(c)}{=}&nH_{\rho_A}.
\end{aligned}$$

We note that (a) holds since the sum is absolutely convergent. (b) follows as $\left\{\left(1-\frac{j}{n}\right)^{r-1}\right\}_{r=1}^{\infty}$ is a geometric series. The equality (c) can be observed by considering Euler's integral representation of the harmonic numbers [28], $H_{\rho_A}=\int_0^1\frac{1-x^{\rho_A}}{1-x}dx$. using the latter we have that

$$\begin{aligned}
H_{\rho_A}&=\int_0^1\frac{1-x^{\rho_A}}{1-x}dx=\int_0^1\frac{1-(1-y)^{\rho_A}}{y}dy\\
&=\sum_{j=1}^{\rho_A}\left(\binom{\rho_A}{j}(-1)^{j+1}\int_0^1 y^{j-1}dy\right)=\sum_{j=1}^{\rho_A}\binom{\rho_A}{j}\frac{(-1)^{j+1}}{j}.
\end{aligned}$$

Thus,

$$\mathbb{E}[\tau_i(\mathcal{C})]=n\cdot\left(H_{\rho_A}+H_{\rho_B}-H_{(\rho_A+\rho_B)}\right),$$

which concludes the proof. □

A direct corollary from Theorem 9.20 is that Example 9.19 can be generalized to any $(n=k+1,k)$ simple parity code $\mathcal{C}$, and for any $i\in[k]$ the expected number of draws to retrieve $\boldsymbol{u}_i$ using $\mathcal{C}$ is exactly k.

Corollary 9.21. *Let $\mathcal{C}$ be the $(n=k+1,k)$ simple parity code (i.e., $\mathbf{X}=(\boldsymbol{u}_1,\ldots,\boldsymbol{u}_k,\sum_{j=1}^{k}\boldsymbol{u}_j)$). Then, for any $i\in[k]$, we have that, $\mathbb{E}[\tau_i(\mathcal{C})]=k$ and $T^{\mathcal{C}}_{\max}=T^{\mathcal{C}}_{avg}=k$.*

The proof of Theorem 9.20 relies on the inclusion-exclusion principle and can be extended to more than two retrieval sets. Since the proof is technical and repeats the same ideas as the ones from Theorem 9.20, it is omitted from the paper.

Corollary 9.22. *Let $\mathcal{C}$ be an (n,k) code and $i\in[k]$. If $\mathcal{D}(i)=\{A_1,A_2,\ldots,A_v\}$ for mutually disjoint retrieval sets, then*

$$\mathbb{E}[\tau_i(\mathcal{C})]=n\cdot\left(\sum_{s=1}^{v}(-1)^{s+1}\sum_{1\le j_1<\cdots<j_s\le v}H_{(|A_{j_1}|+\cdots+|A_{j_s}|)}\right).$$

Corollary 9.21 states that the simple parity code does not improve the value of $T^{\mathcal{C}}_{\max}$. This observation raises the problem of finding codes that indeed improve this parameter, and next we consider MDS codes for this purpose. First, recall that by Lemma 9.15, if no code is used, then we have that $T_{\max} = T_{\text{avg}} = \mathbb{E}[\tau_i] = k$ for any $i \in [k]$. On the other hand, assume $\mathcal{C}$ is a *k-non systematic MDS code* in which the minimal size of a retrieval set, for each of the information strands is k. In other words, any set of less than k encoded strands is not a retrieval set. If $\mathcal{C}$ is used, then in order to retrieve any specific information strand, one should sample a subset of k distinct encoded strands. Hence, by Corollary 9.6, for any $i \in [k]$, we have that, $T^{\mathcal{C}}_{\max} = \mathbb{E}[\tau_i(\mathcal{C})] = \sum_{j=0}^{k-1} \frac{n}{n-j} \approx n \log(\frac{n}{n-k})$, while if $\frac{k}{n} = R$ is a constant, we have that $n \log(\frac{n}{n-k}) = \frac{k}{R} \log(\frac{1}{1-R}) > k$. The next theorem discusses the case where $\mathcal{C}$ is a systematic MDS code and shows that for any such code the expected sample size is exactly k. The proof can be found in Appendix E.

Theorem 9.23. *Let $\mathcal{C}$ be a systematic $[n,k]$ MDS code. For any $i \in [k]$ we have that $\mathbb{E}[\tau_i(\mathcal{C})] = k$ and hence $T^{\mathcal{C}}_{\max} = T^{\mathcal{C}}_{avg} = k$.*

9.6.3 Reducing the Singleton Coverage Depth Below k

In all the codes we studied so far, the expected number of reads to retrieve a single information strand $\boldsymbol{u}_i$, was at least k, which means that these codes do not improve upon the case where no coding is used. Next, we present families of (n,k) codes for which $T^{\mathcal{C}}_{\max} < k$. We start with the following example of an $(8,4)$ code.

Example 9.24. *Let $\mathcal{C}_{(8,4)}$ be the $(8,4)$ code defined as follows. Let $\mathbf{U}_{(8,4)} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3, \boldsymbol{u}_4) \in (\Sigma^\ell)^4$ and let*

$$\mathbf{X}_{(8,4)} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3, \boldsymbol{u}_4, \boldsymbol{u}_1 + \boldsymbol{u}_2, \boldsymbol{u}_2 + \boldsymbol{u}_3, \boldsymbol{u}_3 + \boldsymbol{u}_4, \boldsymbol{u}_4 + \boldsymbol{u}_1) \in (\Sigma^\ell)^8.$$

Denote $\boldsymbol{x}_{i,j} \triangleq \boldsymbol{u}_i + \boldsymbol{u}_j$ and w.l.o.g. assume that we are interested in retrieving $\boldsymbol{u}_1$. It can be verified that

$$\mathcal{D}(1) = \left\{ \begin{array}{c} \{\boldsymbol{u}_1\},\ \{\boldsymbol{u}_2, \boldsymbol{x}_{1,2}\},\ \{\boldsymbol{u}_4, \boldsymbol{x}_{1,4}\}, \\ \{\boldsymbol{u}_3, \boldsymbol{x}_{2,3}, \boldsymbol{x}_{1,2}\},\ \{\boldsymbol{u}_3, \boldsymbol{x}_{3,4}, \boldsymbol{x}_{1,4}\}, \\ \{\boldsymbol{u}_4, \boldsymbol{x}_{3,4}, \boldsymbol{x}_{2,3}, \boldsymbol{x}_{1,2}\},\ \{\boldsymbol{u}_2, \boldsymbol{x}_{3,4}, \boldsymbol{x}_{2,3}, \boldsymbol{x}_{1,4}\} \end{array} \right\},$$

while $\mathcal{D}(1)$ is given with a slight abuse of notation, in which the retrieval sets are given in terms of the encoded strands rather than their indices to simplify the example. Let $\mathcal{E}_{r-1}$ be the random variable that represents the number of unique strands that were sampled in the first $r-1$ draws. Since any set of 6 or more unique strands is a retrieval set of $\boldsymbol{u}_1$, we have that

$$\begin{aligned} P\Big[\tau_1(\mathcal{C}_{(8,4)}) \geq r\Big] &= \sum_{i=1}^{5} P\Big[\tau_1(\mathcal{C}_{(8,4)}) \geq r | \mathcal{E}_{r-1} = i\Big] \cdot P[\mathcal{E}_{r-1} = i] \\ &\quad + P\Big[\tau_1(\mathcal{C}_{(8,4)}) \geq r | \mathcal{E}_{r-1} \geq 6\Big] \cdot P[\mathcal{E}_{r-1} \geq 6] \\ &= \sum_{i=1}^{5} P\Big[\tau_1(\mathcal{C}_{(8,4)}) \geq r | \mathcal{E}_{r-1} = i\Big] \cdot P[\mathcal{E}_{r-1} = i]. \end{aligned}$$

It can be readily verified that $P\Big[\tau_1(\mathcal{C}_{(8,4)}) \geq r | \mathcal{E}_{r-1} = 1\Big] = \frac{7}{8}$. *In case* $\mathcal{E}_{r-1} = 2$, *there are* $\binom{8}{2} = 28$ *different pairs of strands, and since* $\tau_1(\mathcal{C}_{(8,4)}) \geq r$, *we should consider only the pairs from which* $\boldsymbol{u}_1$ *cannot be retrieved. Note that two of the pairs are in* $\mathcal{D}(1)$ *and 7 additional pairs contain* $\boldsymbol{u}_1$. *Hence we have that* $P\Big[\tau_1(\mathcal{C}_{(8,4)}) \geq r | \mathcal{E}_{r-1} = 2\Big] = \frac{28-9}{28} = \frac{19}{28}$. *Similarly, there are* $\binom{8}{3} = 56$ *different triples, from which* $\binom{7}{2} = 21$ *contain* $\boldsymbol{u}_1$, *five more triples contain* $\{\boldsymbol{u}_2, \boldsymbol{x}_{1,2}\}$ *and do not contain* $\boldsymbol{u}_1$, *additional five triples contain* $\{\boldsymbol{u}_3, \boldsymbol{x}_{1,3}\}$ *(and do not contain* $\boldsymbol{u}_1$*), and two more triples are in* $\mathcal{D}(1)$. *That is,* $P\Big[\tau_1(\mathcal{C}_{(8,4)}) \geq r | \mathcal{E}_{r-1} = 3\Big] = \frac{56-21-5-5-2}{56} = \frac{23}{56}$. *Using similar counting techniques, it can be shown that* $P\Big[\tau_1(\mathcal{C}_{(8,4)}) \geq r | \mathcal{E}_{r-1} = 4\Big] = \frac{8}{70}$, *and* $P\Big[\tau_1(\mathcal{C}_{(8,4)}) \geq r | \mathcal{E}_{r-1} = 5\Big] = \frac{1}{56}$. *Furthermore, using the inclusion-exclusion principle, it can be proved that*

$$P[\mathcal{E}_{r-1} = i] = \frac{\binom{8}{i}}{8^{r-1}} \sum_{j=0}^{i-1} \binom{i}{j} (-1)^j (i-j)^{r-1}.$$

By combining all of the above we obtain that

$$\mathbb{E}[\tau_1(\mathcal{C}_{(8,4)})] = \sum_{r=1}^{\infty} P\Big[\tau_1(\mathcal{C}_{(8,4)}) \geq r\Big] = \frac{403}{105} \approx 3.838 = 0.9595k.$$

Example 9.24 can be extended to any integer $k \geq 2$ as follows.

Construction 9.1. *Let* $\mathcal{C}_{(2k,k)}$ *be the* $(n = 2k, k)$ *code such that*

$$\mathbf{U}_{(2k,k)} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k) \in (\Sigma^\ell)^k$$

and

$$\mathbf{X}_{(2k,k)} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k, \boldsymbol{u}_1 + \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{k-1} + \boldsymbol{u}_k, \boldsymbol{u}_k + \boldsymbol{u}_1) \in (\Sigma^\ell)^{2k}.$$

Similarly to Example 9.24, the value $\mathbb{E}[\tau_1(\mathcal{C}_{(2k,k)})]$ can be expressed using the conditional probabilities $P[\tau_1(\mathcal{C}_{(2k,k)}) \geq r | \mathcal{E}_{r-1} = i]$. The evaluation of these conditional probabilities can be done using a recursive formula which is given in the next theorem together with the expected value of $\tau_1(\mathcal{C}_{(2k,k)})$, while the proof appears in Appendix E.

Theorem 9.25. *For any* $k \geq 2$, *and any* $j \in [k]$ *we have that*

$$\mathbb{E}[\tau_j(\mathcal{C}_{(2k,k)})] = 1 + \sum_{i=1}^{2k-3} B(k,i) \cdot \frac{2k}{(2k-i)\binom{2k}{i}},$$

where

$$B(k,i) = \begin{cases} \binom{2k-1}{i} + 2B(k-1,i-1) - B(k-2,i-2) & k \geq 2, i \geq 2 \\ 1 & k \geq 0, i = 0 \\ 2k+1 & k \geq 0, i = 1 \\ 1 & k = 1, i = 2 \\ 0 & k = 0, i \geq 2 \\ 0 & k = 1, i \geq 3 \end{cases}.$$

Even though we did not solve the recursive formula in Theorem 9.25 to obtain an exact value for $\mathbb{E}[\tau_1(\mathcal{C}_{(2k,k)})]$, we used it to calculate $\mathbb{E}[\tau_1(\mathcal{C}_{(2k,k)})]$ for values $2 \leq k \leq 100$ and the results can be found in Figure 9.5. Based on these results we have the following conjecture.

Conjecture 9.26. *For any $k \geq 4$ and any $j \in [k]$, we have that $\mathbb{E}[\tau_j(\mathcal{C}_{(2k,k)})] < k$. Moreover, the ratio $\frac{\mathbb{E}[\tau_j(\mathcal{C}_{(2k,k)})]}{k}$ decreases with k and*

$$\lim_{k\to\infty} \frac{\mathbb{E}[\tau_j(\mathcal{C}_{(2k,k)})]}{k} < 0.9456.$$

The following definition is used in the next theorem.

Definition 9.27. *Given an (n,k) code $\mathcal{C}$ as defined above and an integer $\gamma \geq 1$, we say that a $(\gamma n, \gamma k)$ code $\mathcal{C}^\gamma$ is the* γ-block code of $\mathcal{C}$ *if for an information word*

$$\mathbf{U} = \mathbf{U}_1 \circ \mathbf{U}_2 \cdots \circ \mathbf{U}_\gamma = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k) \circ (\boldsymbol{u}_{k+1}, \ldots, \boldsymbol{u}_{2k}) \circ \cdots \circ (\boldsymbol{u}_{(\gamma-1)k+1}, \ldots, \boldsymbol{u}_{\gamma k}),$$

the corresponding codeword $\mathcal{X}_\gamma$ satisfies,

$$E_{\mathcal{C}^\gamma}(\mathbf{U}) = \mathbf{X} = \mathbf{X}_1 \circ \mathbf{X}_2 \circ \cdots \circ \mathbf{X}_\gamma = E_{\mathcal{C}}(\mathbf{U}_1) \circ E_{\mathcal{C}}(\mathbf{U}_2) \circ \cdots \circ E_{\mathcal{C}}(\mathbf{U}_\gamma),$$

where $E_{\mathcal{C}}$ denotes the encoder of the code $\mathcal{C}$.

In the next theorem, we show that given an (n,k) code $\mathcal{C}$, one can increase k by using a γ-block code $\mathcal{C}^\gamma$, without changing the ratio between the expected number of draws to the number of information strands.

Theorem 9.28. *Let $\mathcal{C}$ be an (n,k) code. For an integer $\gamma \geq 1$, let $\mathcal{C}^\gamma$ be a γ-block code of $\mathcal{C}$. For any $1 \leq i \leq \gamma k$, it holds that, $\mathbb{E}[\tau_i(\mathcal{C}^\gamma)] = \gamma \mathbb{E}[\tau_{i'}(\mathcal{C})]$, where $i' \equiv i \pmod{k}$ and $1 \leq i' \leq k$.*

Proof. For any $r \geq 1$ draws, let us denote by ε_i^r the random variable that governs the number of strands drawn from $\mathbf{X}_s$ (from the r draws), where s is an integer and $u_i \in \mathbf{U}_s$. Then we have

that

$$\begin{aligned}
\mathbb{E}[\tau_i(\mathcal{C}^\gamma)] &= \sum_{r=1}^{\infty} P[\tau_i(\mathcal{C}^\gamma) \geq r] \\
&= \sum_{r=1}^{\infty}\sum_{z=0}^{\infty} P[\varepsilon_i^{r-1} = z] \cdot P\Big[\tau_i(\mathcal{C}^\gamma) \geq r | \varepsilon_i^{r-1} = z\Big] \\
&\overset{(a)}{=} \sum_{r=1}^{\infty}\sum_{z=0}^{r-1} P[\varepsilon_i^{r-1} = z] \cdot P\Big[\tau_i(\mathcal{C}^\gamma) \geq r | \varepsilon_i^{r-1} = z\Big] \\
&= \sum_{r=1}^{\infty}\sum_{z=0}^{r-1} \binom{r-1}{z}\left(\frac{1}{\gamma}\right)^z\left(1-\frac{1}{\gamma}\right)^{r-z-1} \cdot P\Big[\tau_i(\mathcal{C}^\gamma) \geq r | \varepsilon_i^{r-1} = z\Big] \\
&= \sum_{r=1}^{\infty}\sum_{z=0}^{r-1} \binom{r-1}{z}\left(\frac{1}{\gamma}\right)^z\left(1-\frac{1}{\gamma}\right)^{r-z-1} \cdot P[\tau_i(\mathcal{C}) \geq z+1] \\
&= \sum_{z=0}^{\infty} P[\tau_i(\mathcal{C}) \geq z+1] \sum_{r=z+1}^{\infty} \binom{r-1}{z}\left(\frac{1}{\gamma}\right)^z\left(1-\frac{1}{\gamma}\right)^{r-z-1} \\
&= \sum_{z=0}^{\infty} P[\tau_i(\mathcal{C}) \geq z+1] \sum_{r=z}^{\infty} \binom{r}{z}\left(\frac{1}{\gamma}\right)^z\left(1-\frac{1}{\gamma}\right)^{r-z} \\
&\overset{(b)}{=} \sum_{z=0}^{\infty} P[\tau_i(\mathcal{C}) \geq z+1] \cdot \gamma \\
&= \sum_{z=1}^{\infty} P[\tau_i(\mathcal{C}) \geq z] \cdot \gamma = \gamma \mathbb{E}[\tau_i(\mathcal{C})],
\end{aligned}$$

where equality (a) follows from the fact that the probability to collect $z > r-1$ unique strands from $\mathbf{X}_s$, using only $r-1$ draws is zero for any integer s, i.e., $P[\varepsilon_i^{r-1} = z] = 0$. To see that equality (b) holds, recall that $\sum_{r=0}^{\infty} x^r = \frac{1}{1-x}$, and by taking the derivative of the latter z times we get

$$\sum_{r=z}^{\infty} r \cdot (r-1) \cdots (r-z+1) x^{r-z} = \frac{z!}{(1-x)^{z+1}},$$

which is equivalent to

$$\sum_{r=z}^{\infty} \binom{r}{z} x^{r-z} = \frac{1}{(1-x)^{z+1}}.$$

Lastly, by substituting $x = 1 - \frac{1}{\gamma}$, equality (a) follows. □

Theorem 9.28 implies that given an (n,k) code $\mathcal{C}$, that achieves good results in terms of minimizing the expressions $\frac{\mathbb{E}[\tau_i(\mathcal{C})]}{k}$, for $i \in [k]$, it is possible to construct an infinte family of fixed-rate codes $\{\mathcal{C}^\gamma\}_{\gamma=1}^{\infty}$, such that for any integer $\gamma \geq 1$, $\mathcal{C}^\gamma$ is an $(\gamma n, \gamma k)$ code and for any $i_\gamma \in [\gamma k]$ there exists $i \in [k]$, such that

$$\frac{\mathbb{E}[\tau_{i_\gamma}(\mathcal{C}^\gamma)]}{\gamma k} = \frac{\mathbb{E}[\tau_i(\mathcal{C})]}{k}.$$

That is, for any integer $\gamma \geq 1$, the code $\mathcal{C}^\gamma$ has the *same behavior* as the code $\mathcal{C}$ in terms of minimizing the normalized expected singleton coverage depth. Hence, combining Example 9.24 and Theorem 9.28 leads to the following corollary.

Corollary 9.29. *For any integer $\gamma \geq 1$, let $\mathcal{C}^\gamma_{(8\gamma,4\gamma)}$ be the γ-block code of $\mathcal{C} = \mathcal{C}_{(8,4)}$ (see Example 9.24). For any $i_\gamma \in [\gamma k]$, where $k = 4$, we have that*

$$\mathbb{E}\Big[\tau_{i_\gamma}(\mathcal{C}^\gamma_{(8\gamma,4\gamma)})\Big] = T^{\mathcal{C}^\gamma_{(8\gamma,4\gamma)}}_{max} = T^{\mathcal{C}^\gamma_{(8\gamma,4\gamma)}}_{avg} = 0.9595\gamma k.$$

Note that our numerical computations of the expression in Theorem 9.25 imply that the value $\frac{\mathbb{E}[\tau_i(\mathcal{C}_{(2k,k)})]}{k}$ decreases with k, for $2 \leq k \leq 100$. In particular, for $k > 3$, $\frac{\mathbb{E}[\tau_i(\mathcal{C}_{(2k,k)})]}{k} \leq 0.9456$ and thus by Theorem 9.28 it is possible to construct codes that improve upon the result in Corollary 9.29 for infinite values of k.

Next, we demonstrate that the value $\frac{\mathbb{E}[\tau_i(\mathcal{C})]}{k}$ can be further reduced, by letting the rates of our codes vanish.

Construction 9.2. *Let n be an integer, $p \in (0,1)$, and assume for simplicity that np is an integer that is dividable by k. Additionally, let $\mathcal{C}^{MDS}_{n,k,p}$ be a $[n(1-p)+k, k]$ systematic MDS code. We define the (n,k) code $\mathcal{C}^k_{n,p}$ as follows. For $\mathbf{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k) \in (\Sigma^\ell)^k$, let*

$$(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{(1-p)n}) \in (\Sigma^\ell)^{n(1-p)+k}$$

be the encoding of $\mathbf{U}$ using the encoder of $\mathcal{C}^{MDS}_{n,k,p}$. Then,

$$\mathbf{X} = (\underbrace{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_1}_{\frac{pn}{k} \text{ times}}, \underbrace{\boldsymbol{u}_2, \ldots, \boldsymbol{u}_2}_{\frac{pn}{k} \text{ times}}, \ldots, \underbrace{\boldsymbol{u}_k, \ldots, \boldsymbol{u}_k}_{\frac{pn}{k} \text{ times}}, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{(1-p)n}) \in (\Sigma^\ell)^n.$$

Theorem 9.30. *For $k = 2, 3$, there exists $p_2, p_3 \in (0,1)$, such that for any information strand $i \in [k]$, we have that,*

$$\mathbb{E}[\tau_i(\mathcal{C}^2_{n,p_2})] \approx 1.83 = 0.9143k,$$

and

$$\mathbb{E}[\tau_i(\mathcal{C}^3_{n,p_3})] \approx 2.67 = 0.89k.$$

Proof. We prove the claim only for $k = 2$, while the proof for $k = 3$ relies on the exact same ideas. Assume w.l.o.g. that we want to retrieve $\boldsymbol{u}_1$. For simplicity of the analysis, also assume that $\mathbf{X}$ contains two information stands $\boldsymbol{u}_1, \boldsymbol{u}_2$ (without multiplicity), and that each of them can be drawn with probability $\frac{p}{2}$. First note that since $(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{(1-p)n})$ belongs to a $[(1-p)n+2, 2]$ MDS code, any two distinct strands form a retrieval set for $\boldsymbol{u}_1$, and hence the only case in which we didn't retrieve $\boldsymbol{u}_1$ in r draws, is when we draw the same strand (which is not $\boldsymbol{u}_1$) r times.

- $\tau_1(\mathcal{C}^2_{n,p}) = 1$ only in case we draw $\boldsymbol{u}_1$ in the first draw which happens with probability $\frac{p}{2}$.

- $\tau_1(\mathcal{C}^2_{n,p}) = r$ for $r \geq 2$ only if the first $r-1$ draws are of the strand $\boldsymbol{x} \neq \boldsymbol{u}_1$ and the last draw is of a different strand. Hence we have that

$$P[\tau_1(\mathcal{C}^2_{n,p}) = r] = \left(\frac{p}{2}\right)^{r-1}\left(1-\frac{p}{2}\right) + (1-p)n \cdot \left(\frac{1}{n}\right)^{r-1}\left(1-\frac{1}{n}\right).$$

Thus,

$$\begin{aligned}
\mathbb{E}[\tau_1(\mathcal{C}^2_{n,p})] &= \sum_{r=1}^{\infty} P[\tau_i(\mathcal{C}_{n,p}) = r] \cdot r \\
&= \frac{p}{2} + \sum_{r=2}^{\infty} r\left(\left(\frac{p}{2}\right)^{r-1}\left(1-\frac{p}{2}\right) + (1-p)n \cdot \left(\frac{1}{n}\right)^{r-1}\left(1-\frac{1}{n}\right)\right) \\
&= \frac{p}{2} + \left(1-\frac{p}{2}\right)\sum_{r=2}^{\infty} r\left(\frac{p}{2}\right)^{r-1} + (1-p)\left(1-\frac{1}{n}\right)\sum_{r=2}^{\infty} r\left(\frac{1}{n}\right)^{r-2}.
\end{aligned}$$

For n large enough $(1-\frac{1}{n})\sum_{r=2}^{\infty} r\left(\frac{1}{n}\right)^{r-2} \approx 2$ and hence for n large enough we have that

$$\begin{aligned}
\mathbb{E}[\tau_1(\mathcal{C}^2_{n,p})] &\approx \frac{p}{2} + \left(1-\frac{p}{2}\right)\sum_{r=2}^{\infty} r\left(\frac{p}{2}\right)^{r-1} + 2(1-p) \\
&= \frac{p}{2} + \left(1-\frac{p}{2}\right)\frac{p(4-p)}{(2-p)^2} + 2(1-p) \\
&= \frac{p}{2} + \frac{p(4-p)}{2(2-p)} + 2(1-p).
\end{aligned}$$

This expression is minimized when $p = 2-\sqrt{2}$ and in this case we have

$$\frac{p}{2} + \frac{p(4-p)}{2(2-p)} + 2(1-p) \approx 1.83.$$

Note that even though the optimal p is irrational, since the latter function is continuous for $p \in (0,1)$, we can get as close as we want to this optimum value. Hence, we have that

$$\mathbb{E}[\tau_1(\mathcal{C}^2_{n,p})] \approx 1.83 = 0.9143k.$$

□

Combining Theorem 9.28 and Theorem 9.30 leads to the following corollary.

Corollary 9.31. *Let $p_2, p_3 \in (0,1)$ be the constants from Theorem 9.30. For any integer $\gamma \geq 1$, let $\mathcal{C}^{2,\gamma}_{n,p_2}$ be the $(n\gamma, 2\gamma)$ γ-block code of $\mathcal{C} = \mathcal{C}^2_{n,p_2}$, and similarly let $\mathcal{C}^{3,\gamma}_{n,p_3}$ be the $(n\gamma, 3\gamma)$ γ-block code of $\mathcal{C} = \mathcal{C}^3_{n,p_3}$ (see Definition 9.27). For any $i_2 \in [2\gamma]$, and $i_3 \in [3\gamma]$ we have that*

$$\mathbb{E}[\tau_{i_2}(\mathcal{C}^{2,\gamma}_{n,p_2})] = T^{\mathcal{C}^{2,\gamma}_{n,p_2}}_{max} = T^{\mathcal{C}^{2,\gamma}_{n,p_2}}_{avg} \approx 1.83\gamma = 0.9143 \cdot (2\gamma),$$

and

$$\mathbb{E}[\tau_{i_3}(\mathcal{C}^{3,\gamma}_{n,p_3})] = T^{\mathcal{C}^{3,\gamma}_{n,p_3}}_{max} = T^{\mathcal{C}^{3,\gamma}_{n,p_3}}_{avg} \approx 2.67\gamma = 0.89 \cdot (3\gamma).$$

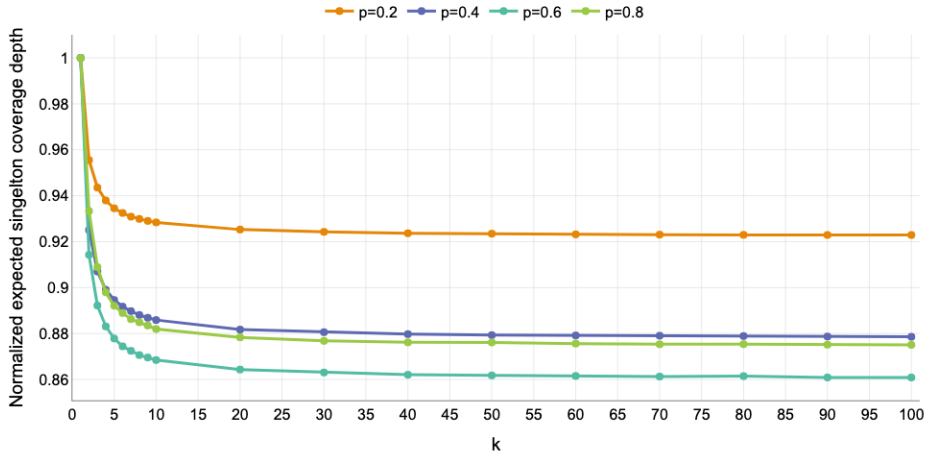


Figure 9.3: Approximated values of $\mathbb{E}[\tau_i(\mathcal{C}^k_{n,p})]$ (Construction 9.2) for different values of $p \in \{0.2, 0.4, 0.6, 0.8\}$ as a function of $k \in \{1, 2, \ldots, 10\} \cup \{20, 30, \ldots, 100\}$, where $n = 10^8$. The approximated values were obtained empirically by $10,000,000$ computer simulations per any pair of values of k and p. The presented results are normalized by k.

The evaluation of $\mathbb{E}[\tau_i(\mathcal{C}^k_{n,p})]$ for $k > 3$ can be done using the same technique, however, it becomes less elegant and we do not attempt to evaluate the latter expression rigorously. Nevertheless, we did try to gain a better understanding of the behavior of these codes by computer simulations as follows. Each simulation was done while fixing $k \in \{1, \ldots, 10\} \cup \{20, 30, \ldots 100\}$ and $p \in \{0.2, 0.4, 0.6, 0.8\}$. For each pair of k and p we started by encoding k information strands with an $[\lfloor (1-p)n \rfloor + k, k]$ MDS code $\mathcal{C}^{\text{MDS}}_{n,k,p}$, from which we constructed the (n, k) code $\mathcal{C}^k_{n,p}$ (see Construction 9.2). Then, we simulated the sampling process by picking a single strand at each draw (with an equal probability of $\frac{1}{n}$). The simulation stops whenever we can recover $\boldsymbol{u}_1$. We repeated this process 10^7 times for each pair of k and p and plotted the mean number of the required draws, which is an empirical approximation of $\mathbb{E}[\tau_i(\mathcal{C}^k_{n,p})]$. Our simulations imply that for most of the tested values of k, the optimal value of p is around 0.6. Furthermore, it can be seen that $\mathbb{E}[\tau_i(\mathcal{C}^k_{n,p})]$ decreases as k increases. Finally, it should be noted that even though such codes are not applicable, they allow us to gain insights about the achievable values of $\mathbb{E}[\tau_1(\mathcal{C})]$.

9.6.4 Lower Bounds

This section concludes with lower bounds on the value of $\mathbb{E}[\tau_i(\mathcal{C})]$.

Lemma 9.32. *For any (n, k) code $\mathcal{C}$, $T^{\mathcal{C}}_{\max} \geq \frac{k+1}{2}$.*

Proof. Assume the word $\mathbf{U}$ was encoded to the codeword $\mathbf{X}$. Every sequence of reads can be expressed as a vector $\boldsymbol{v} \in [n]^*$, and for every such a $\boldsymbol{v}$, denote by $n_i(\boldsymbol{v})$, for $i \in [k]$, the minimum read index h which allows retrieving the i-th information strand $\boldsymbol{u}_i$. The key

intuition behind our approach is that each new sample collected during the sequence of reading the strands allows us to recover at most one new information strand. Hence,

$$\sum_{i=1}^{k} n_i(\boldsymbol{v}) = n_1(\boldsymbol{v}) + n_2(\boldsymbol{v}) + \cdots + n_k(\boldsymbol{v}) \geq \sum_{i=1}^{k} i = k(k+1)/2.$$

Hence, it follows that, $\sum_{i=1}^{k} \tau_i(\mathcal{C}) \geq k(k+1)/2$ and therefore

$$\mathbb{E}\left[\sum_{i=1}^{k} \tau_i(\mathcal{C})\right] = \sum_{i=1}^{k} \mathbb{E}[\tau_i(\mathcal{C})] \geq k(k+1)/2.$$

In particular, there exists $i \in [k]$ for which $\mathbb{E}[\tau_i(\mathcal{C})] \geq \frac{k+1}{2}$, i.e., $T^{\mathcal{C}}_{\max} \geq \frac{k+1}{2}$. □

Even though the bound in Lemma 9.32 holds for any code $\mathcal{C}$, it appears that in most cases the bound is not tight. To obtain a tighter lower bound on $T^{\mathcal{C}}_{\max}$ in the next theorem we also consider the rate of the code $\mathcal{C}$.

Theorem 9.33. *Let $\mathcal{C}$ be an (n,k) code. It holds that*

$$T^{\mathcal{C}}_{\max} \geq \frac{n}{k} \cdot \sum_{i=0}^{k} \frac{k-i}{n-i} = n - \frac{n(n-k)}{k} \cdot (H_n - H_{n-k}).$$

Proof. Let us use the same notations as in the proof of Lemma 9.32. Additionally, denote by $t_i(v)$ the time to collect the i-th new sample (after collecting the previous one). Clearly, we have that

$$\sum_{i=1}^{k} \tau_i(\mathcal{C}) = \sum_{i=1}^{k} n_i(v) \geq \sum_{i=1}^{k} \sum_{j=1}^{i} t_j(v).$$

Define $t_j(\mathcal{C})$ to be the random variable that governs the time to collect the j-th new sample (after collecting the previous one). Hence,

$$\sum_{i=1}^{k} \mathbb{E}[\tau_i(\mathcal{C})] = \mathbb{E}\left[\sum_{i=1}^{k} \tau_i(\mathcal{C})\right] \geq \mathbb{E}\left[\sum_{i=1}^{k} \sum_{j=1}^{i} t_j(\mathcal{C})\right] = \sum_{i=1}^{k} \sum_{j=1}^{i} \mathbb{E}\big[t_j(\mathcal{C})\big].$$

Note that for any $j \in [k]$, we have that $t_j(\mathcal{C})$ is a geometric random variable with success probability $p_j = \frac{n-(j-1)}{n}$ and so $\mathbb{E}[t_j(\mathcal{C})] = \frac{n}{n-(j-1)}$, and

$$\begin{aligned}
\sum_{i=1}^{k} \mathbb{E}[\tau_i(\mathcal{C})] \geq \sum_{i=1}^{k} \sum_{j=1}^{i} \mathbb{E}\big[t_j(\mathcal{C})\big] &= \sum_{i=1}^{k} \sum_{j=1}^{i} \frac{n}{n-(j-1)} \\
&= n \sum_{i=1}^{k} \left(\frac{1}{n-i+1} + \frac{1}{n-i+2} + \ldots + \frac{1}{n} \right) \\
&= n \left(\frac{k}{n} + \frac{k-1}{n-1} + \ldots + \frac{1}{n-k+1} \right) = n \sum_{i=0}^{k-1} \frac{k-i}{n-i}.
\end{aligned}$$

For any $i \in [k]$, we have that

$$\frac{k-i}{n-i} = \frac{k}{n} - \left(1 - \frac{k}{n}\right)\frac{i}{n-i},$$

which implies that

$$\begin{aligned} n\sum_{i=0}^{k}\frac{k-i}{n-i} &= n\sum_{i=0}^{k-1}\left(\frac{k}{n} - \left(1-\frac{k}{n}\right)\frac{i}{n-i}\right) \\ &= k^2 - n\left(1-\frac{k}{n}\right)\sum_{i=0}^{k-1}\frac{i}{n-i} \\ &= k^2 - (n-k)\sum_{i=0}^{k-1}\left(\frac{n}{n-i} - 1\right) \\ &= k^2 + k(n-k) - n(n-k)\sum_{i=0}^{k-1}\frac{1}{n-i} \\ &= nk - n(n-k)(H_n - H_{n-k}). \end{aligned}$$

Hence we have that

$$\frac{1}{k}\sum_{i=1}^{k}\mathbb{E}[\tau_i(\mathcal{C})] \geq \frac{n}{k}\cdot\sum_{i=0}^{k}\frac{k-i}{n-i} = n - \frac{n(n-k)}{k}\cdot(H_n - H_{n-k}).$$

In particular, there exists $i \in [k]$ for which $\mathbb{E}[\tau_i(\mathcal{C})] \geq \frac{n}{k}\cdot\sum_{i=0}^{k}\frac{k-i}{n-i} = n - \frac{n(n-k)}{k}\cdot(H_n - H_{n-k})$, i.e., $T^{\mathcal{C}}_{\max} \geq \frac{n}{k}\cdot\sum_{i=0}^{k}\frac{k-i}{n-i} = n - \frac{n(n-k)}{k}\cdot(H_n - H_{n-k})$. □

Lastly, we conclude with the following lemma.

Corollary 9.34. *Let $0 < R < 1$ and consider a sequence of codes $\{\mathcal{C}_i\}_{i=1}^{\infty}$ with parameters (n_i, k_i) such that for any i, $n_i < n_{i+1}$, and $R = \frac{k_i}{n_i}$. It holds that,*

$$\lim_{i\to\infty}\frac{T^{\mathcal{C}_i}_{\max}}{k_i} \geq \left(\frac{1}{R} + \frac{1-R}{R^2}\cdot\log(1-R)\right).$$

That is, for any $\varepsilon > 0$, there exists i large enough (i.e., n_i, k_i large enough) such that,

$$T^{\mathcal{C}_i}_{\max} \geq k_i\left(\frac{1}{R} + \frac{1-R}{R^2}\cdot\log(1-R)\right) - \varepsilon.$$

Proof. From Theorem 9.33, for any (n,k) code $\mathcal{C}$, we have that

$$T^{\mathcal{C}}_{\max} \geq n - \frac{n(n-k)}{k}\cdot(H_n - H_{n-k}).$$

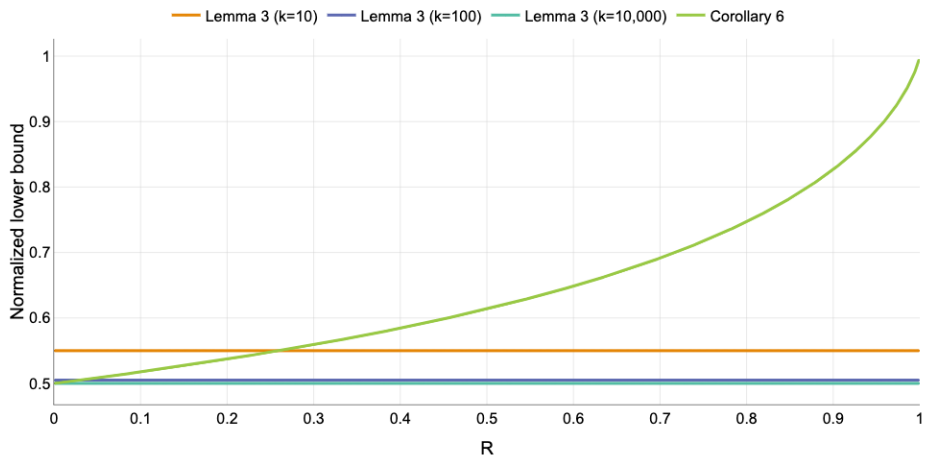


Figure 9.4: Comparison of the lower bounds (Lemma 9.32 and Corollary 9.34) as a function of the rate $R = \frac{k}{n}$. The presented results are normalized by k.

Thus, we have that

$$\begin{aligned}
\lim_{i\to\infty} \frac{T^{\mathcal{C}_i}_{\max}}{k_i} &\geq \lim_{i\to\infty} \frac{1}{k_i}\left(n_i - \frac{n_i(n_i-k_i)}{k_i}\cdot\left(H_{n_i} - H_{n_i-k_i}\right)\right) \\
&= \lim_{i\to\infty} \frac{n_i}{k_i}\left(1 - \frac{n_i-k_i}{k_i}\cdot\left(H_{n_i} - H_{n_i-k_i}\right)\right) \\
&= \lim_{i\to\infty} \frac{1}{R} - \frac{(1-R)}{R^2}\left(H_{n_i} - H_{n_i-k_i}\right) \\
&= \frac{1}{R} - \frac{1-R}{R^2}\log\left(\frac{1}{1-R}\right).
\end{aligned}$$

It can be verified that in this case if R approaches zero, one, then the latter expression approaches $\frac{1}{2}, 1$, respectively.

□

Figure 9.4 presents a comparison between the lower bounds of Lemma 9.32 and Corollary 9.34 as a function of the code rate $R = \frac{k}{n}$. As can be seen in the figure, in most cases, the bound in Corollary 9.34 is tighter than the one from Lemma 9.32. More than that, the code rate from which the bound in Corollary 9.34 is tighter than the bound from Lemma 9.32 decreases with k.

Finally, we give in Figure 9.5 a comparison of the normalized expected singleton coverage depth for different codes with rate of exactly $R = 0.5$. It can be seen in the figure that the k-non systematic MDS code achieves the worst results, while the code in Theorem 9.25 achieves the best results, which are roughly 55% lower than the k-non systematic MDS code and roughly 10% lower than a systematic MDS code. To offer a better understanding of these results, the lower bounds discussed in Lemma 9.32 and Corollary 9.34 are also given in the figure.

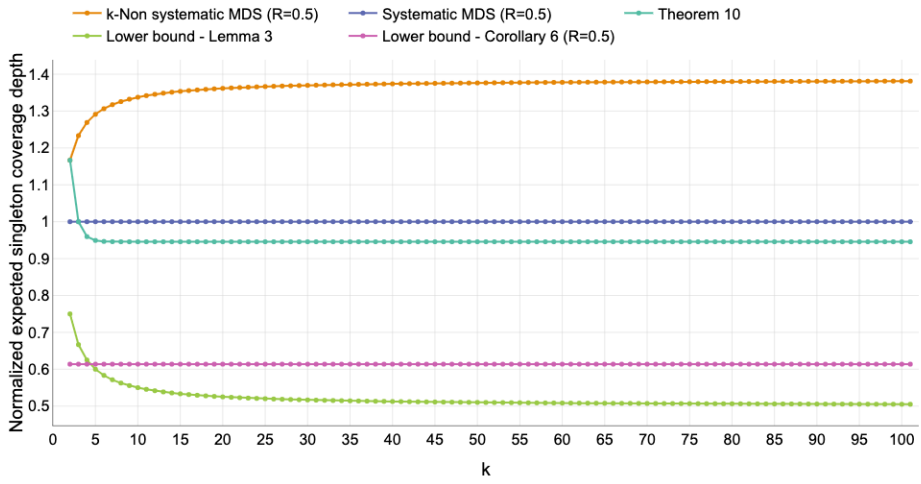


Figure 9.5: Comparison of the normalized expected singleton coverage depth for code with rate $R = 0.5$.

9.7 Conclusion

In this paper, we have introduced and extensively investigated the novel problem of DNA coverage depth, aiming to reduce sequencing costs and latency while ensuring high-accuracy retrieval. Our contributions encompass the MDS coverage depth problem, demonstrating the superiority of MDS codes in the noiseless channel. For noisy channels, we proved several bounds on the probability of successfully retrieving the information for a given sample size. Additionally, we have explored the singleton coverage depth problem, revealing insights into code properties and retrieval times, as well as presenting code constructions that can improve the retrieval time. These findings collectively provide a foundational framework for designing efficient and reliable DNA storage systems, with potential implications for advancing the field.

Nonetheless, future research should address the diverse challenges posed by different noise models, investigate coding schemes beyond MDS codes, and extend the coverage depth problem for additional scenarios. Several possible directions and open problems are listed below.

1) Extend the results presented in this paper with respect to Problem 9.1 from a uniform distribution to additional channel distributions $\boldsymbol{p}$; e.g. the normal distribution.

2) In this work, the noisy channel was modeled by a parameter t, under the assumption that retrieval succeeds with probability 1 given t or more noisy copies and fails otherwise. A very relevant extension to this noise model is to consider the more realistic behavior of the channel, in which the success probability can increase or decrease as a function of the number of noisy copies, i.e., as a function of the cluster size.

3) Define and study the coverage depth random access problem for the case in which a subset of size greater than one should be retrieved. This can be considered for arbitrary subsets of the information strands or for pre-defined subsets, that represent units of

information (e.g. files).

4) Study the coverage depth random access problem under the assumption of noisy channel and/or channel with non-uniform distribution.

Acknowledgement

The authors thank John M. Hoffman for raising up the real-world necessity of minimizing the coverage depth and understanding how it can be done with coding and to Zohar Yakhini for helpful discussions about the theoretical definition of the model. The authors also thank Tomer Cohen for analyzing the code presented in Construction 9.1. Lastly, the authors thank Ron M. Roth for suggesting the current version of the proof of Theorem 9.5, which is more elegant than its original version.

Bibliography

[1] L. Anavy, I. Vaknin, O.Atar, R. Amit and Z. Yakhini, "Data storage in DNA with fewer synthesis cycles using composite DNA letters". *Nature Biotechnology,* vol. 37, no. 10, pp. 1229–1236, 2019.

[2] J. L. Banal, et al., "Random access DNA memory using Boolean search in an archival file storage system," *Nature Material*, vol. 20, no. 9, pp. 1272–1280, 2021.

[3] D. Bar-Lev, O. Sabary, R. Gabrys, and E. Yaakobi, "Cover your bases: How to minimize the sequencing coverage in DNA storage systems," *2023 IEEE International Symposium on Information Theory (ISIT)*, 2023, pp. 370–375.

[4] V. Bhardwaj, P. A. Pevzner, C. Rashtchian, and Y. Safonova, "Trace reconstruction problems in computational biology," *IEEE Trans. on Information Theory*, vol. 67, no. 6, pp. 3295–3314, 2021.

[5] H. P. J. Buermans and J. T. D. Dunnen, "Next generation sequencing technology: Advances and applications," *Biochimica et Biophysica Acta (BBA)-Molecular Basis of Disease*, vol. 1842, no. 10, pp. 1932–1941, 2014.

[6] S. Chandak et al., "Improved read/write cost tradeoff in DNA-based data storage using LDPC codes," *Annual Allerton Conference on Communication, Control, and Computing*, 2019.

[7] I. Chatzigeorgiou, "Bounds on the Lambert Function and their application to the outage analysis of user cooperation," *IEEE Communications Letters*, vol. 17, no. 8, pp. 1505–1508, 2013.

[8] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *The Annals of Mathematical Statistics*, vol. 23, no. 4, pp. 493–507, 1952.

[9] I. Csiszar, "ℓ-divergence geometry of probability distributions and minimization problems," *The Annal of Probabilty*, vol. 3, no. 1, pp. 146–158, 1975.

[10] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth, "On the LambertW function," *Advances in Computational Mathematics*, vol. 5, pp. 329–359, 1996.

[11] P. Erdős, and A. Rényi, "On a classical problem of probability theory," *Magyar Tud. Akad. Mat. Kutató Int. Közl* vol. 6, no. 1, pp. 215–220, 1961.

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

[13] A. Fazeli, A. Vardy, and E. Yaakobi, "Codes for distributed PIR with optimal storage overhead," *2015 IEEE International Symposium on Information Theory (ISIT)*, 2015, pp. 2852–2856.

[14] W. Feller, *An introduction to probability theory and its applications*, John Wiley & Sons, vol. 1, 2nd edition, 1967.

[15] P. Flajolet, D. Gardy, and L. Thimonier, "Birthday paradox, coupon collectors, caching algorithms and self-organizing search," *Discrete Applied Mathematics*, vol. 39, no. 3, pp. 207-229, 1992.

[16] G. B. Arfken, H. J. Weber, and F. E. Harris, "Chapter 13 - Gamma function," in *Mathematical Methods for Physicists (Seventh Edition)*, Academic Press, pp. 599–641, https://www.sciencedirect.com/topics/mathematics/digamma-function, 2013.

[17] R. Heckel, G. Mikutis, and R. N. Grass, "A Characterization of the DNA data storage channel," *Scientific Reports*, vol. 9, no. 1, p. 9663, 2019.

[18] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *The collected works of Wassily Hoeffding*, pp. 409–426, 1994.

[19] P. Huang, E. Yaakobi, H. Uchikawa and P. H. Siegel, "Linear locally repairable codes with availability," *2015 IEEE International Symposium on Information Theory (ISIT)*, 2015, pp. 1871-1875.

[20] B. Lau et al., "Magnetic DNA random access memory with nanopore readouts and exponentially-scaled combinatorial addressing," *Scientific Reports*, vol. 13, no. 1, p. 8514, 2023.

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

[22] E. M. LeProust et al., "Synthesis of high-quality libraries of long (150mer) oligonucleotides by a novel depurination controlled process," *Nucleic Acids Research*, vol. 38, no. 8, pp. 2522–2540, 2010.

[23] D. J. Newman, "The double dixie cup problem," *The American Mathematical Monthly*, vol. 67, no. 1, pp. 58-61, 1960.

[24] L. Organick et al., "Random access in large-scale DNA data storage," *Nature Biotechnology*, vol. 36, no. 3, pp. 242–248, 2018.

[25] A. N. Philippou, C. Georghiou, G. N. Philippou, "A generalized geometric distribution and some of its properties," *Statistics & Probability Letters*, vol. 1, no. 4, pp.171–175, 1983.

[26] D. S. Papailiopoulos, and A. G. Dimakis, "Locally repairable codes," *IEEE Transaction on Information Theory*, vol. 60, no. 10, pp. 5843–5855, 2014.

[27] J. Rydning. "Worldwide IDC Global DataSphere Forecast, 2022–2026: Enterprise Organizations Driving Most of the Data Growth," *International Data Corporation (IDC)*, 2022.

[28] E. Sandifer, "How Euler did it". *Washington: Mathematics Association of America*, 2007.

[29] J. Sima, N. Raviv, and J. Bruck, "On coding over sliced information," *IEEE Transactions on Information Theory*, vol. 67, no. 5, pp. 2793–2807, 2021.

[30] I. Shomorony and R. Heckel, "Information-theoretic foundations of DNA data storage," *Foundations and Trends in Communications and Information Theory* vol. 19, no. 1, pp 1–106, 2022.

[31] A. Vardy and E. Yaakobi, "Private information retrieval without storage overhead: Coding instead of replication," t *IEEE Journal on Selected Areas in Information Theory*, vol. 4, pp. 286–301, 2023.

[32] Y. Wang, Y. Zhao, A. Bollas, Y. Wang, and K. F. Au, "Nanopore sequencing technology, bioinformatics and applications," *Nature Biotechnology*, vol. 39, no. 11, pp. 1348–1365, 2021.

[33] C. Winston, L. Organick, D. Ward, L. Ceze, K. Strauss, and Y.-J. Chen, "Combinatorial PCR method for efficient, selective oligo retrieval from complex oligo pools", *ACS Synth. Biol.*, vol. 11, no. 5, pp. 1727–1734, 2022.

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

[35] S. M. H. T. Yazdi, H. M. Kiah, E. Garcia-Ruiz, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: trends and methods," *IEEE Transactions on Molecular, Biological and Multi-Scale Communications* vol. 1, no. 3, pp. 230–248, Sep. 2015.

[36] S. M. H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, Random-access DNA-based storage system," *Scientific Reports*, vol. 5, no. 1, pp. 1–10, 2015.

[37] White paper by DNA Data Storage Alliance, "Preserving our digital legacy: An introduction to DNA data storage," a publication of *DNA Data Storage Aliance*, 2021.

Appendix A

Claim 9.35. *For $n > 16$, it holds that,*

$$\sum_{j=0}^{t-1} \binom{r}{j} \left(\frac{1}{n}\right)^j \left(1-\frac{1}{n}\right)^{r-j} \leq t \cdot \binom{r}{t-1} \left(\frac{1}{n}\right)^{t-1} \left(1-\frac{1}{n}\right)^{r-(t-1)}$$

Proof. To prove the claim, we show that for $0 \leq j \leq t-1$,

$$\binom{r}{j} \left(\frac{1}{n}\right)^j \left(1-\frac{1}{n}\right)^{r-j} \leq \binom{r}{j+1} \left(\frac{1}{n}\right)^{j+1} \left(1-\frac{1}{n}\right)^{r-(j+1)}.$$

The latter can be proved by showing the following equivalent inequality

$$n \leq \frac{r-j}{j+1} \left(1-\frac{1}{n}\right)^{-1}.$$

The expression $\frac{r-j}{j+1}\left(1-\frac{1}{n}\right)^{-1}$ is minimized at $j = t-1$ (considering only $j \in \{0,1,\ldots,t-1\}$), thus it is enough to show that, $n \leq \frac{r-t+1}{t}\left(1-\frac{1}{n}\right)^{-1} = \frac{r-t+1}{t}\left(\frac{n}{n-1}\right)$, which follows if

$$r = r(n,k,t) \geq t(n-1) + (t-1).$$

Lastly, it can be verified that $r(n,k,t) \geq t(n-1)+(t-1)$ for any integers $n > 16$ and $t > 1$. □

Proof of Theorem 9.9

Denote $r_f \triangleq r_f(n,k,t)$. Similarly to the proof of Theorem 9.8, when n is large enough, the probability that urn i has at most $t-1$ balls after r_f draws is denoted by $P(z_i(n,r_f) \leq t-1)$, where $z_i(n,r_f)$ is a defined as in the proof of Theorem 9.8. Thus, the probability is given by,

$$\begin{aligned} P(z_i(n,r_f) \leq t-1) &\leq 3t \cdot \left(\frac{r_f}{n}\right)^{t-1} \left(1-\frac{1}{n}\right)^{n\left(\frac{r_f}{n}-\frac{t-1}{n}\right)} \\ &\leq 6t^t \frac{(2\cdot f(n))^{t-1}}{e^{t\cdot f(n)}} \cdot (1-R) \end{aligned}$$

Now let us define a random variable Y as the number of urns with less than t balls. As in the proof of Theorem 9.8, we have that,

$$P(E_t^{(r_f)}) = P(Y \geq n-k+1) \leq 6t^t \frac{(2\cdot f(n))^{t-1}}{e^{t\cdot f(n)}}(1-R),$$

where the last inequality follows from Markov's inequality.

Appendix B

Theorem 9.36. *For any $\varepsilon > 0$, $n > e^{\frac{6t \cdot 2^{t-1}}{\varepsilon}} \geq 15$, and any integer $h \geq 1$, we have that,*

$$P[\nu_t(n,k) > h \cdot r(n,k,t)] < \varepsilon \cdot \frac{h^{t-1}}{\log^{t(h-1)}(n)}.$$

Proof. Denote $r \triangleq r(n,k,t)$ and recall that within the context of the urn problem (see Subsection 9.3.2), the random variable $\nu_t(n,k)$ denotes the number of balls (or rounds) necessary to guarantee that we have a set of k urns where each urn has at least t balls.

If r balls are drawn, we show that the probability that there are at least k urns each with at least t balls is approaching one when n grows. Analogous to the approach used in the previous section, we will show that if the number of balls thrown is at least r, then the probability to have at most $n-k+1$ urns which are *not* filled with t balls is approaching zero.

The approach leveraged in this section is inspired by a technique first employed by Erdős and Rényi in [11]. First, we define the following event.

$E_t^{(r)}$**:** After r rounds, there exists a set S_t, of $n-k+1$ urns, each containing less than t balls.

Next, we show that the probability of $E_t^{(r)}$ approaches zero when n is large. We first define $z_i(n,r)$ for $1 \leq i \leq n$, as a random variable that governs the number of balls in the i-th urn, after r draws. For n large enough, the probability that urn i has at most $t-1$ balls after $h \cdot r$ draws is denoted by $P(z_i(n,r) \leq t-1)$ and is given by,

$$\begin{aligned} P(z_i(n,h\cdot r) \leq t-1) &= \sum_{j=0}^{t-1} \binom{h\cdot r}{j} \left(\frac{1}{n}\right)^j \left(1-\frac{1}{n}\right)^{h\cdot r - j} \\ &\leq t \cdot \binom{h\cdot r}{t-1} \left(\frac{1}{n}\right)^{t-1} \left(1-\frac{1}{n}\right)^{h\cdot r-(t-1)} \\ &\leq t \cdot \left(\frac{h\cdot r\cdot e}{t-1}\right)^{t-1} \left(\frac{1}{n}\right)^{t-1} \left(1-\frac{1}{n}\right)^{h\cdot r-(t-1)}, \end{aligned}$$

where the last inequality follows from the fact that $\binom{h\cdot r}{t-1} \leq \frac{hre}{t-1}^{t-1}$. Note that $(\frac{e}{t-1})^{t-1} < 3$, for $t > 1$. Thus,

$$P(z_i(n,r) \leq t-1) \leq 3t \cdot \left(\frac{hr}{n}\right)^{t-1} \left(1-\frac{1}{n}\right)^{n\left(\frac{r}{n}-\frac{t-1}{n}\right)}.$$

We have that,

$$P(z_i(n,r) \leq t-1) \leq 3t \cdot \left(h \log\left(\frac{n}{n-k}\right) + ht \log\log(n) + 2h\log(t+1)\right)^{t-1} \left(e^{\left(\frac{-hr}{n} + \frac{t-1}{n}\right)}\right)$$
$$\leq 3t \cdot (2h\log n)^{t-1} \left(\frac{n-k}{n}\right)^h \left(\frac{1}{\log^{ht} n}\right) \left(\frac{1}{(t+1)^{2h}}\right) e^{\frac{t-1}{n}}$$
$$= 3t \cdot \frac{e^{\frac{t-1}{n}}}{(t+1)^{2h}} \cdot \frac{(2h\log n)^{t-1}}{\log^{ht}(n)} \cdot \left(\frac{n-k}{n}\right)^h$$
$$= 3t \cdot \frac{e^{\frac{t-1}{n}}}{(t+1)^{2h}} \cdot \frac{(2h)^{t-1}}{\log^{t(h-1)+1}(n)} \cdot \left(\frac{n-k}{n}\right)^h,$$

where the second inequality holds since for n large enough $h\log(\frac{n}{n-k}) + th\log\log(n) + 2h\log(t+1) \leq (2h\log n)$. It should be noted that for $n > t$, we have that $3t \cdot \frac{e^{\frac{t-1}{n}}}{(t+1)^{2h}} \leq 6t$, and hence,

$$P(z_i(n,r) \leq t-1) \leq 6t \cdot \frac{(2h)^{t-1}}{\log^{t(h-1)+1}(n)} \cdot \left(\frac{n-k}{n}\right)^h.$$

Now let us define a random variable Y as the number of urns with less than t balls. From the linearity of expectation, regardless if the urns are independent or not, the expected number of urns that have less than t balls is,

$$\lim_{n\to\infty} \mathbb{E}[Y] = \lim_{n\to\infty} \Sigma_{i=1}^{n} \mathbb{E}[z_i(n,r)]$$
$$= \lim_{n\to\infty} nP(z_i(n,r) \leq t-1) \leq \lim_{n\to\infty} (n-k)\left(\frac{n-k}{n}\right)^{h-1} \cdot 6t \cdot \frac{(2h)^{t-1}}{\log^{t(h-1)+1}(n)}$$

Note that

$$P(E_{t+1}^{(r)}) = P(Y \geq n-k+1)$$

Using Markov's inequality with $n-k+1$ as the parameter we can conclude that,

$$P(Y \geq n-k+1) \leq \left(\frac{n-k}{n}\right)^{h-1} 6t \cdot \frac{(2h)^{t-1}}{\log^{t(h-1)+1}(n)}.$$

Let us denote $\varepsilon^* = 6t \cdot \frac{2^{t-1}}{\log(n)}$, then we get that,

$$P(Y \geq n-k+1) \leq \varepsilon^* \left(\frac{n-k}{n}\right)^{h-1} \cdot \frac{(h)^{t-1}}{\log^{t(h-1)}(n)} \leq \varepsilon^* \cdot \frac{(h)^{t-1}}{\log^{t(h-1)}(n)}.$$

Hence, we get that $P(E_t^{(r)}) \to 0$ for n large enough which implies the statement in the theorem. □

Appendix C

In this appendix, we prove Theorem 9.13. The proof is partially based on Claim 9.12 which is proven next.

Proof of Claim 9.12:

Recall that by (9.9), for $r \geq nt$ we have that

$$E[X^{(r)}] \leq ne^{-(t-1)\log_2 \frac{n(t-1)}{r} - (r-(t-1))\log_2 \frac{(r-(t-1))n}{r(n-1)}}.$$

Hence, a sufficient condition for $E[X^{(r)}] \leq n-k = n(1-R)$, it that

$$e^{-(t-1)\log_2 \frac{n(t-1)}{r} - (r-(t-1))\log_2 \frac{(r-(t-1))n}{r(n-1)}} \leq (1-R). \tag{9.12}$$

Note that

$$\begin{aligned} e^{-(t-1)\log_2 \frac{n(t-1)}{r} - (r-(t-1))\log_2 \frac{(r-(t-1))n}{r(n-1)}} &= e^{-\frac{t-1}{\ln 2}\ln(\frac{n(t-1)}{r}) - \frac{(r-(t-1))}{\ln 2}\ln\left(\frac{(r-(t-1))n}{r(n-1)}\right)} \\ &= \left(\frac{n(t-1)}{r}\right)^{-\frac{t-1}{\ln 2}} \left(\frac{(r-(t-1))n}{r(n-1)}\right)^{-\frac{(r-(t-1))}{\ln 2}} \\ &= \left(\frac{r}{n(t-1)}\right)^{\frac{t-1}{\ln 2}} \left(\frac{rn-r}{rn-(t-1)n}\right)^{\frac{r-(t-1)}{\ln 2}}, \end{aligned}$$

and by denoting $r = \beta n(t-1)$, for some $\beta \geq 1$ we can rewrite the sufficient condition in (9.12) as follows.

$$\beta\left(1 - \frac{(\beta-1)}{\beta n - 1}\right)^{\beta n - 1} \leq (1-R)^{\frac{\ln 2}{t-1}}. \tag{9.13}$$

By the definition of e, for any constant β we have that $\left(1 - \frac{(\beta-1)}{\beta n-1}\right)^{\beta n-1} \leq e^{-(\beta-1)}$. Hence, if $\beta e^{-\beta} \leq \frac{1}{e}(1-R)^{\frac{\ln 2}{t-1}}$ holds than (9.13) also holds.

By the assumption,

$$-\frac{r}{n(t-1)} e^{-\frac{r}{n(t-1)}} = -\beta e^{-\beta} \geq -\frac{1}{e}(1-R)^{\frac{\ln 2}{t-1}}$$

or equivalently

$$\beta e^{-\beta} \leq \frac{1}{e}(1-R)^{\frac{\ln 2}{t-1}}$$

which completes the proof. □

The following two claims are known results related to the Lambert W function and are given as part of the proof of Theorem 9.13.

Claim 9.37. *[10, Section IV] For any real numbers $-\frac{1}{e} \leq x < 0$ and y, the equation $ye^y = x$ has exactly two solutions which are given by $y = W_0(x)$ and $y = W_{-1}(x)$, where W_0 and W_{-1} are branches of the Lambert W function.*

Claim 9.38. *[7, Theorem 1] For any $u > 0$ we have that*

$$-1 - \sqrt{2u} - u < W_{-1}(-e^{-u-1}) < -1 - \sqrt{2u} - \frac{2}{3}u.$$

Proof of Theorem 9.13

Denote $x = \frac{1}{e}(1-R)^{\frac{\ln 2}{t-1}}$ and $y = \frac{r}{n(t-1)}$, by Claim 9.37, the equation

$$-\frac{r}{n(t-1)}e^{-\frac{r}{n(t-1)}} = -ye^{-y} = -x = -\frac{1}{e}(1-R)^{\frac{\ln 2}{t-1}}$$

has exactly two solutions which are $y = -W_0(-x)$ and $y = -W_{-1}(-x)$. Note that for any $y \geq 1$ the function $-ye^{-y}$ is continuous and monotonically increasing with y. Hence, for any given $R = \frac{k}{n}$ only the branch W_{-1} is relevant. This implies that for $r \geq n(t-1)$, Equation (9.10) holds if and only if $y \geq -W_{-1}(-x)$.

By Claim 9.38, we know that $-W_{-1}(-e^{-u-1}) < 1 + \sqrt{2u} + u$ for any $u > 0$. We can rewrite $-x$ as

$$-x = -\frac{1}{e}(1-R)^{\frac{\ln 2}{t-1}} = -e^{\frac{\ln 2}{t-1}\ln(1-R)-1}$$

with $u = -\frac{\ln 2}{t-1}\ln(1-R) > 0$ to obtain

$$\begin{aligned} -W_{-1}(-x) &< 1 + \sqrt{2u} + u \\ &= 1 + \sqrt{-\frac{2\ln 2}{t-1}\ln(1-R)} - \frac{\ln 2}{t-1}\ln(1-R) \end{aligned}$$

Hence, a sufficient condition for $\mathbb{E}[X^{(r)}] \leq n - k$ is that

$$y = \frac{r}{n(t-1)} \geq 1 + \sqrt{-\frac{2\ln 2}{t-1}\ln(1-R)} - \frac{\ln 2}{t-1}\ln(1-R),$$

or equivalently,

$$r \geq n(t-1) - n\ln 2\ln(1-R) + n(t-1)\sqrt{-\frac{2\ln 2}{t-1}\ln(1-R)}.$$

□

Appendix D

Proof of Theorem 9.14:

We denote by ω_i the probability of collecting an error-free new strand, given that $i-1$ strands were collected. In this case, $\omega_i = \alpha^{\frac{n-(i-1)}{n}} = \alpha^{\frac{n-i+1}{n}}$. We let t_i be the time to collect a new

error-free strand, given $i-1$ such strands were already sampled. Since t_i is geometric random variable it holds that $t_i = \frac{1}{\omega_i}$. Thus, from the linearity of expectation, we have that

$$\begin{aligned} E[\omega_\alpha(n,k)] &= \mathbb{E}[t_1 + t_2 + \dots t_n] - \mathbb{E}[t_k + t_{k+1} + \dots t_n] \\ &= \mathbb{E}[t_1] + \mathbb{E}[t_2] + \dots \mathbb{E}[t_n] - \mathbb{E}[t_k] + \mathbb{E}[t_{k+1}] + \dots \mathbb{E}[t_n] \\ &= \frac{n}{\alpha}\left(\frac{1}{1} + \frac{1}{2} + \dots + \frac{1}{n}\right) - \frac{n}{\alpha}\left(\frac{1}{1} + \frac{1}{2} + \dots + \frac{1}{n-k}\right) \\ &= \frac{n}{\alpha}(H_n - H_{n-k}) \end{aligned}$$

□

Appendix E

Proof of Theorem 9.23:

Let $W_{k,n}$ be the random variable that represents the number of samples needed to obtain k distinct coupons where each draw is taken from a pool of n total coupons. We denote by $W_{k,n}(x)$ the generating function for $W_{k,n}$. For $x < \frac{1}{1-\frac{n-k+1}{n}} = \frac{n}{k-1}$ it is known [25] that,

$$W_{k,n}(x) = \sum_{r=0}^{x} P[W_{k,n} = r] \cdot x^r = \prod_{i=1}^{k} \frac{(n-i+1)x}{n-(i-1)x}.$$

Additionally, let $V_{r,n}$ be the random variable that represents the number of distinct coupons in the first r draws, where each coupon is taken from a pool of n total coupons. Note that

$$\begin{aligned} P[W_{k,n} = r] &= P[V_{r-1,n} = k-1] \cdot P[V_{r,n} = k | V_{r-1,n} = k-1] \\ &= \frac{n-(k-1)}{n} P[V_{r-1,n} = k-1], \end{aligned}$$

and hence,

$$P[V_{r-1,n} = k-1] = \frac{n}{n-(k-1)} P[W_{k,n} = r]. \tag{9.14}$$

We let $D_{k,i,n}$ denote the random variable that represents the required number of draws to obtain k distinct coupons or to retrieve coupon i (whichever occurs first), where each draw is taken from a pool of n total coupons. We denote by $D_{k,i,n}(x)$ the generating function for $D_{k,i,n}$. To this end we define $D^{(j)}_{k,i,n}$ for $0 \leq j \leq k-1$, to be the random variable that represents the number of samples needed to obtain j distinct coupons (each not equal to the i-th coupon), followed by the i-th coupon. Additionally, let $D^{(k)}_{k,i,n}$, be the random variable that represents the number of samples needed to obtain k distinct coupons (each not equal to the i-th coupon).

It holds that,

$$P[D^{(k)}_{k,i,n} = r] = \left(1 - \frac{1}{n}\right)^r \cdot P[W_{k,n-1} = r],$$

and

$$\begin{aligned} D^{(k)}_{k,i,n}(x) &= \sum_{r=0}^{\infty} x^r \cdot P[D^{(k)}_{k,i,n} = r] \\ &= \sum_{r=0}^{\infty} x^r \cdot \left(1 - \frac{1}{n}\right)^r \cdot P[W_{k,n-1} = r] \\ &= W_{k,n-1}\left(\left(1 - \frac{1}{n}\right)x\right) \\ &= \prod_{\ell=1}^{k} \frac{(n-\ell)(1-\frac{1}{n})x}{n-1-(\ell-1)(1-\frac{1}{n})x}. \end{aligned}$$

For $1 \leq j \leq k-1$, using (9.14), we have that

$$\begin{aligned} P[D^{(j)}_{k,i,n} = r] &= \left(1 - \frac{1}{n}\right)^{r-1} \cdot \left(\frac{1}{n}\right) \cdot P[V_{r-1,n-1} = j] \\ &= \left(1 - \frac{1}{n}\right)^{r-1} \cdot \left(\frac{1}{n}\right) \cdot \frac{n-1}{n-1-j} \cdot P[W_{j+1,n-1} = r] \end{aligned}$$

and,

$$\begin{aligned} D^{(j)}_{k,i,n}(x) &= \sum_{r=0}^{\infty} x^r \cdot P[D^{(j)}_{k,i,n} = r] \\ &= \sum_{r=0}^{\infty} x^r \cdot \left(1 - \frac{1}{n}\right)^{r-1} \cdot \left(\frac{1}{n}\right) \cdot \frac{n-1}{n-1-j} \cdot P[W_{j+1,n-1} = r] \\ &= \frac{n-1}{n-1-j} \cdot \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{-1} \cdot \sum_{r=0}^{\infty} x^r \cdot \left(1 - \frac{1}{n}\right)^r \cdot P[W_{j+1,n-1} = r] \\ &= \frac{1}{n-1-j} \cdot W_{j+1,n-1}\left(\left(1 - \frac{1}{n}\right)x\right) \\ &= \frac{1}{n-1-j} \prod_{\ell=1}^{j+1} \frac{(n-\ell)\left(1-\frac{1}{n}\right)x}{n-1-(\ell-1)\left(1-\frac{1}{n}\right)x}. \end{aligned}$$

Note that, $P[D^{(0)}_{k,i,n} = r] = \frac{1}{n}$ if $r = 1$ and otherwise $P[D^{(0)}_{k,i,n} = r] = 0$. Therefore, we

have that $D^{(0)}_{k,i,n}(x) = \frac{x}{n}$. Next, we present $D_{k,i,n}(x)$ as a function of $D^{(j)}_{k,i,n}$ for $0 \leq j \leq k$.

$$\begin{aligned} D_{k,i,n}(x) &= \sum_{r=0}^{\infty} x^r \cdot P[D_{k,i,n} = r] \\ &= \sum_{r=0}^{\infty} x^r \sum_{j=0}^{k} P[D^{(j)}_{k,i,n} = r] \\ &= \sum_{j=0}^{k} \sum_{r=0}^{\infty} x^r \cdot P[D^{(j)}_{k,i,n} = r] \\ &= \sum_{j=0}^{k} D^{(j)}_{k,i,n}(x). \end{aligned}$$

From the above, it can be derived that,

$$\begin{aligned} \mathbb{E}[D_{k,i,n}] = D'_{k,i,n}(1) &= \frac{1}{n} + \sum_{j=1}^{k-1} \frac{n(n-(j+1))(\psi(-n) - \psi(j+1-n))}{n(n-(j+1))} \\ &+ \frac{n(n-k)(\psi(-n) - \psi(k-n))}{n} \\ &= \frac{1}{n} + \sum_{j=1}^{k-1} (\psi(-n) - \psi(j+1-n)) + (n-k)(\psi(-n) - \psi(k-n)), \end{aligned}$$

where $\psi(z) \triangleq \int_0^{\infty} \left(\frac{e^{-t}}{t} - \frac{e^{-zt}}{1-e^{-t}} \right) dt$ is the digamma function. In [16], it is claimed that for any $z \in \mathbb{C}$ and $j \in \mathbb{N}$, we have that, $\psi(z+j) = \psi(z) + \sum_{h=1}^{j} \frac{1}{z+h-1}$, which implies that, $\psi(-n) - \psi(j-n) = H_n - H_{n-j}$.

Hence,

$$\begin{aligned} \mathbb{E}[\tau_i(\mathcal{C})] = \mathbb{E}[D_{k,i,n}] &= \frac{1}{n} + \sum_{j=1}^{k-1} \left(H_n - H_{n-j-1} \right) + (n-k)(H_n - H_{n-k}) \\ &= \frac{1}{n} + (n-1)H_n - (n-k)H_{n-k} - \sum_{j=1}^{k-1} H_{n-j-1} \\ &= \frac{1}{n} + (n-1)H_n - (n-k)H_{n-k} - \left(\sum_{j=1}^{n-2} H_j - \sum_{j=1}^{n-k-1} H_j \right) \\ &\overset{(a)}{=} \frac{1}{n} + (n-1)H_n - (n-k)H_{n-k} - ((n-1)(H_{n-1} - 1 - (n-k)(H_{n-k} - 1))) \\ &= \frac{1}{n} + (n-1)(H_n - H_{n-1} + 1) - (n-k) \\ &= \frac{1}{n} + \frac{n-1}{n} + (n-1) - n + k \\ &= k, \end{aligned}$$

where (a) follows since for any integer $n > 0$ we have that $\sum_{j=1}^{n} H_j = (n+1)H_n - n$.

Appendix F

Proof of Theorem 9.25:

Let $\mathcal{C}_k$ be the systematic $(2k,k)$ code that is defined by $\mathbf{U}_k = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k)$ and

$$\mathbf{X}_k = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k, \boldsymbol{u}_1 + \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{k-1} + \boldsymbol{u}_k, \boldsymbol{u}_k + \boldsymbol{u}_1).$$

Similarly to Example 9.24, we have that

$$P[\tau_1(\mathcal{C}) \geq r] = \sum_{i=1}^{2k-3} P[\tau_1(\mathcal{C}) \geq r | \mathcal{E}_{r-1} = i] \cdot P[\mathcal{E}_{r-1} = i],$$

and

$$P[\mathcal{E}_r = i] = \frac{\binom{2k}{i}}{(2k)^r} \sum_{j=0}^{i} \binom{i}{j} (-1)^{i-j} (i-j)^r.$$

Since $P[\tau_1(\mathcal{C}) \geq r | \mathcal{E}_{r-1} = i]$ does not depend on r, let us denote $P_i = P[\tau_1(\mathcal{C}) \geq r | \mathcal{E}_{r-1} = i]$. We have that

$$\begin{aligned}
\mathbb{E}[\tau_1(\mathcal{C})] &= \sum_{r=0}^{\infty} P[\tau_1(\mathcal{C}) \geq r] = 1 + \sum_{r=1}^{\infty} \sum_{i=1}^{2k-3} P_i \cdot P[\mathcal{E}_{r-1} = i] \\
&= 1 + \sum_{i=1}^{2k-3} P_i \sum_{r=1}^{\infty} P[\mathcal{E}_{r-1} = i] \\
&= 1 + \sum_{i=1}^{2k-3} P_i \sum_{r=1}^{\infty} \binom{2k}{i} \sum_{j=0}^{i} (-1)^{i-j} \binom{i}{j} \left(\frac{j}{2k}\right)^{r-1} \\
&= 1 + \sum_{i=1}^{2k-3} P_i \binom{2k}{i} \sum_{j=0}^{i} (-1)^{i-j} \binom{i}{j} \sum_{r=0}^{\infty} \left(\frac{j}{2k}\right)^{r} \\
&= 1 + \sum_{i=1}^{2k-3} P_i \binom{2k}{i} (-1)^i \sum_{j=0}^{i} (-1)^j \binom{i}{j} \cdot \frac{2k}{2k-j} \\
&= 1 + \sum_{i=1}^{2k-3} P_i \cdot 2k \cdot \binom{2k}{i} \cdot (-1)^i \cdot \frac{(-1)^i}{(2k-i)\binom{2k}{i}} \\
&= 1 + \sum_{i=1}^{2k-3} P_i \cdot \frac{2k}{(2k-i)}.
\end{aligned}$$

Hence, our goal is to calculate P_i. Let $A(2k-1,i)$ be the number of options to draw $r-1$ strands such that $\boldsymbol{u}_1$ cannot be recovered from this set of draws, knowing that the set of different encoded strands that were drawn is of size exactly i. Then, we have that $P_i = \frac{A(2k-1,i)}{\binom{2k}{i}}$.

To present a recursive expression for the values $A(2k-1,i)$, we describe an equivalent way to represent the options that contribute to $A(2k-1,i)$. Let $G_k = (V,E)$ be the directed graph with the $2k-1$ nodes that correspond to the symbols in $\mathcal{X}$ excluding $\boldsymbol{u}_1$. The set E consists of the following edges:

- For each $2 \leq j \leq k-1$, the vertex $\boldsymbol{u}_j + \boldsymbol{u}_{j+1}$ has four outgoing edges. Two green outgoing edges to the nodes $\boldsymbol{u}_j$ and $\boldsymbol{u}_{j-1} + \boldsymbol{u}_j$, and two blue outgoing edges to the nodes $\boldsymbol{u}_{j+1}$ and $\boldsymbol{u}_{j+1} + \boldsymbol{u}_{j+2}$ (where $\boldsymbol{u}_{j+2} = \boldsymbol{u}_1$ if $j = k-1$).
- There are two blue outgoing edges from $\boldsymbol{u}_1 + \boldsymbol{u}_2$, to the nodes $\boldsymbol{u}_2$ and $\boldsymbol{u}_2 + \boldsymbol{u}_3$.
- There are two green outgoing edges from $\boldsymbol{u}_k + \boldsymbol{u}_0$, to the nodes $\boldsymbol{u}_k$ and $\boldsymbol{u}_{k-1} + \boldsymbol{u}_k$.

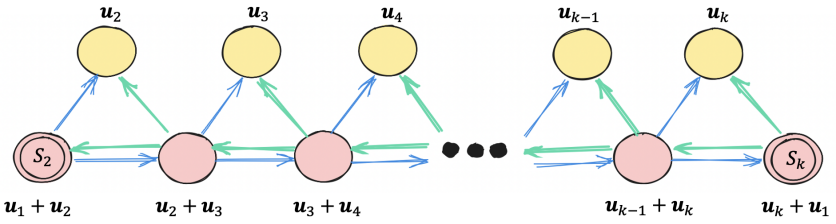


Figure 9.6: Schematic description of G_k

Denote the nodes $\boldsymbol{u}_1 + \boldsymbol{u}_2$ and $\boldsymbol{u}_k + \boldsymbol{u}_1$ by S_2 and S_k, respectively. Additionally, denote the nodes $\boldsymbol{u}_j$, for $2 \leq j \leq k$ by *ending nodes*. For a set $J \subseteq [2k] \backslash \{1\}$, let $G_k^{(J)}$ be the subgraph of G_k that contains all the nodes that correspond to J (considering their locations in $\mathbf{X}$). Note that any set $J \subseteq [2k]$ of size i is not a retrieval set of $\boldsymbol{u}_1$ if and only if the subgraph of $G_k^{(J)}$, does not contain a monochromatic path from S_2 or S_k to one of the ending nodes (if S_2, S_k is not in $G_k^{(J)}$, we say that there is no such path from S_2, S_k, respectively). Hence, $A(2k-1,i)$ is equal to the number of subgraphs $G_k^{(J)}$ of G_k, such that $J \subseteq [2k] \backslash \{1\}$ and $G_k^{(J)}$ does not contain a monochromatic path from S_2 or S_k to one of the ending nodes. Denote the nodes of $G_k^{(J)}$ by V' and consider the following cases.

1) If $S_2, S_k \notin V'$ then any such a subgraph $G_k^{(J)}$ cannot contain a valid monochromatic path and there are $\binom{2k-3}{i}$ such subgraphs.
2) If $S_2 \in V'$ then we have that $\boldsymbol{u}_2 \notin V'$ and there are $A(2k-3,i-1)$ such sub-graphs.
3) If $S_k \in V'$ then we have that $\boldsymbol{u}_k \notin V'$ and there are $A(2k-3,i-1)$ such sub-graphs.
4) If $S_2, S_k \in V'$ then we have that $\boldsymbol{u}_2, \boldsymbol{u}_k \notin V'$ and there are $A(2k-5,i-2)$ such subgraphs.

Thus,

$$A(2k-1,i) = \binom{2k-3}{i} + 2A(2k-3,i-1) - A(2k-5,i-2).$$

By denoting $B(k,i) = A(2k+1,i)$, we can write the latter as

$$B(k,i) = \binom{2k-1}{i} + 2B(k-1,i-1) - B(k-2,i-2),$$

for any $k \geq 2, i \geq 2$, and for all $k \geq 0$ we have that $B(k,0) = 1$ and $B(k,1) = 2k+1$. Additionally $A(1,2) = 1$, for $i \geq 2$ we have $A(0,i) = 0$ and for $i \geq 3$ we have $B(1,i) = 0$.

Thus, we have that

$$\mathbb{E}[\tau_1(\mathcal{C})] = 1 + \sum_{i=1}^{2k-3} B(k,i) \cdot \frac{2k}{(2k-i)\binom{2k}{i}}.$$

Discussion

Chapter 10

Discussion

The rapid growth of digital data and the limitations of traditional storage methods have necessitated exploration into alternative storage solutions, with DNA-based storage emerging as a promising candidate. My research explores the exciting field of DNA storage systems, investigating both theoretical foundations and practical implementations. The works presented here encompass a wide range of topics, including error behavior analysis, coding strategies, and practical system design. The dissertation is based on my papers, which were published in and submitted to leading journals and conferences, primarily in the fields of coding theory and information theory.

In the first part of this thesis, we focused on understanding deletion and insertion errors, which are predominant in DNA storage systems. By investigating the properties of deletion and insertion balls, we provided theoretical foundations for designing efficient error correction codes tailored to the unique error characteristics of DNA storage. Our research in this area, as presented in Part I, contributes to a better understanding of error correction in DNA storage systems.

Moving forward, our attention shifted to coding techniques specifically tailored for DNA storage. In Section 1.2, we discussed the importance of efficient coding schemes in addressing challenges such as DNA fragmentation and enforcing specific constraints like the almost-balanced GC content. Our research in this area, as detailed in Part II, provides novel insights and solutions for encoding and decoding information in DNA storage systems, ensuring data integrity and reliability.

Finally, in Part III, we explored the practical considerations and challenges in transitioning DNA storage from a theoretical concept to a real-world application. Despite the immense potential of DNA storage, practical implementation still faces hurdles such as high costs, limited throughput, and computational complexities. Our research in Part III addresses these challenges by proposing innovative solutions such as DNAformer, which combines deep learning and coding theory techniques to enhance error correction capabilities. Furthermore, we delved into minimizing sequencing coverage depth, addressing the challenge of reducing sequencing costs and latency while ensuring high-accuracy retrieval and exploring alternative representation methods, such as DNA labeling, to reduce the reliance on costly synthesis processes.

10.1 Deletion and Insertion Errors

The exploration of coding for DNA-based storage systems necessitates a thorough investigation into the prevalent challenges posed by deletion and insertion errors, which can significantly impact data integrity. Part I of this thesis presents a comprehensive inquiry into the fundamental combinatorial objects underlying these errors, aiming to unravel their complexities and lay the groundwork for developing robust strategies for mitigation.

Central to our investigation is the FLL metric, which is the metric to use when analyzing channels with the same number of deletion and insertion errors. The FLL distance between two words of the same length, $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{Z}_q^n$ is one-half the minimum number of insertions and deletions that is required to transform a word $\boldsymbol{x}_1$ into $\boldsymbol{x}_2$. Equivalently, the FLL distance between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is equal to the length of the longest (shortest) common subsequence (supersequence) of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.

Our study in Chapter 3 initially focused on balls under the FLL metric, aiming to analyze the size of such balls. First, we tackled the minimum size of balls under the FLL metric and showed that for any $0 \leq t \leq n$ and any alphabet size $q \geq 2$ we have that

$$\min_{\boldsymbol{x} \in \mathbb{Z}_q^n} |\mathcal{L}_t(\boldsymbol{x})| = \sum_{i=0}^{t} \binom{n}{i} (q-1)^i,$$

and the minimum is obtained only by the balls centered at $\boldsymbol{x} = \sigma^n$ for any $\sigma \in \mathbb{Z}_q$.

Then, we turned to study the maximum size of balls under the FLL metric, where we focused on balls with radius one and considered the binary and the non-binary cases separately. For $q > 2$, we showed that the maximum FLL 1-balls are the balls centered at $\boldsymbol{x} \in \mathbb{Z}_q^n$, such that the number of runs in $\boldsymbol{x}$ is n (i.e., any two consecutive symbols are different) and $x_i \neq x_{i+2}$ for all $1 \leq i \leq n-2$. In addition, the maximum size of an FLL 1-ball is,

$$\max_{\boldsymbol{x} \in \mathbb{Z}_q^n} |\mathcal{L}_1(\boldsymbol{x})| = n^2(q-1) - n + 2.$$

Surprisingly, the analysis of the binary case revealed itself as much more complex, and through rigorous analysis, we found that for any integer $n \geq 1$ the maximum FLL 1-balls are the balls centered at the α-balanced sequences of length n, where α is (one of) the integer(s) closest to $\frac{1}{2}\sqrt{1+2n}$. In addition, the size of the maximum FLL 1-balls is given by

$$\begin{aligned}\max_{\boldsymbol{x} \in \mathbb{Z}_2^n} \{|\mathcal{L}_1(\boldsymbol{x})|\} = n^2 - n\alpha + \alpha + 1 - \frac{k}{2}\Big(\Big\lceil \frac{n}{\alpha} \Big\rceil - 1\Big)\Big(\Big\lceil \frac{n}{\alpha} \Big\rceil - 2\Big) \\ - \frac{\alpha - k}{2}\Big(\Big\lceil \frac{n}{\alpha} \Big\rceil - 2\Big)\Big(\Big\lceil \frac{n}{\alpha} \Big\rceil - 3\Big),\end{aligned}$$

where $k \equiv n \pmod{\alpha}$ and $1 \leq k \leq \alpha$.

The average size of a ball with a radius one under the FLL metric was also addressed and we showed that

$$\mathop{\mathbb{E}}_{\boldsymbol{x} \in \mathbb{Z}_q^n}[|\mathcal{L}_1(\boldsymbol{x})|] = n^2\left(q + \frac{1}{q} - 2\right) - \frac{n}{q} - \frac{(q-1)(q-2)}{q^2} + 3 - \frac{3}{q} + \frac{2}{q^2} + \frac{q^n - 1}{q^n(q-1)}.$$

In addition to balls under the FLL metric, we extended our investigation and considered anticodes under this metric. This includes the identification of several fundamental properties of (maximal) anticodes under the FLL metric and the analysis of the minimum and maximum sizes of maximal binary anticodes with diameter one. Namely, we proved that the size of any maximal binary anticode with diameter one is between 4 and $n+1$. The latter can be extended to a non-binary alphabet, and while the minimum size of a maximal anticode with a diameter one is 4 for any alphabet size q, the maximum size of a maximal anticode with diameter one is $n(q-1)+1$.

Recently, based on these results, Wang and Wang [75] extended the analysis of 1-FLL balls by proving that the size of the 1-FLL balls is highly concentrated around its mean using Azuma's inequality [2]. Later, He and Ye [30] explored the size of the 2-FLL balls, providing valuable insights into the expectation and concentration for balls with a larger radius. While these findings shed more light on the size of balls under the FLL metric, there are plenty of open questions regarding the sizes of the FLL balls, including the maximum size, the average size, the size distribution, and the properties of words that maximize the size of the balls for larger radii. Another future direction is to study bounds on codes that can correct insertions and deletions using existing results regarding the sizes of FLL balls.

Moving forward, our attention in Chapter 4 turns to the intersection between deletion and insertion balls—an aspect of coding for DNA storage systems that has not been extensively addressed in prior research. In the insertion-deletion intersection problem, we let $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \Sigma_q^*$ and $n \in \mathbb{N}$ such that $|\boldsymbol{y}_1| \leq n \leq |\boldsymbol{y}_2|$ and the goal is to find the set of all words $\boldsymbol{x} \in \Sigma_q^n$ such that $\boldsymbol{x}$ is both, a supersequence of $\boldsymbol{y}_1$ and a subsequence of $\boldsymbol{y}_2$. That is, the goal is to find the following set

$$\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) \triangleq \mathcal{I}_{n-|\boldsymbol{y}_1|}(\boldsymbol{y}_1) \cap \mathcal{D}_{|\boldsymbol{y}_2|-n}(\boldsymbol{y}_2).$$

The intersection of balls plays a pivotal role in various applications, including sequence reconstruction and list decoding. First, we study the maximum size of a t_1-insertion ball and a t_2-deletion ball for the binary case and showed that for integers $0 \leq t_1 \leq n$, $0 \leq t_2$, and $q = 2$, the size of the maximum intersection is

$$\max_{\boldsymbol{y}_1 \in \Sigma_2^{n-t_1}, \boldsymbol{y}_2 \in \Sigma_2^{n+t_2}} |\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)| = \sum_{i=0}^{\min\{t_1,t_2\}} \binom{n}{i}.$$

Afterwords, we focused on the case where $t_1 = t_2 = 1$, and considered a general alphabet size q. By rigorously analyzing different types of pairs of sequences, we offered valuable insights into the intersection of single-insertion and single-deletion balls. In particular, we provided an explicit expression for the size of $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$ for any two sequences $\boldsymbol{y}_1, \boldsymbol{y}_2$ such that $|\boldsymbol{y}_1| = n-1$ and $|\boldsymbol{y}_2| = n+1$. Moreover, the study presents an efficient algorithm for computing the intersection of insertion and deletion balls, offering a practical tool that can be used in algorithms for the DNA reconstruction problem.

While these findings collectively advance our comprehension of deletion and insertion errors, there remains much to be explored. This extends beyond merely understanding the combinatorial structures inherent to insertion and deletion errors. It encompasses a broader endeavor of leveraging this knowledge to explore the attainable rates of deletion and insertions correcting codes and the capacity of corresponding channels. Moreover, it entails the design of codes

capable of achieving these bounds. As articulated by Mitzenmacher [51], "More generally, channels with synchronization errors, including both insertions and deletions as well as more general timing errors, are simply not adequately understood by current theory. Given the near-complete knowledge we have regarding channels with erasures and errors, in terms of both the channel capacity and codes that can nearly achieve capacity, our lack of understanding about channels with synchronization errors is truly remarkable." This highlights the persistent necessity for continued research and innovation in error correction mechanisms tailored for DNA storage systems.

10.2 Coding for DNA Storage

The exploration of coding techniques tailored specifically for DNA storage systems represents a crucial step toward enhancing data reliability and integrity in such systems. Part II delves into the challenges of coding for DNA storage, with a focus on two key areas: adversarial torn-paper codes and optimal almost-balanced sequences.

The research on adversarial torn-paper codes, which appears in Chapter 5, aimed to address challenges posed by the torn-paper channel model in DNA storage systems. We extended the previously studied probabilistic setting to worst-case scenarios, introducing the concept of the adversarial torn-paper channel. In this model, an information string is adversarially segmented into non-overlapping substrings, with each substring's length falling between $L_{\min}$ and $L_{\max}$. Our analysis revealed that the channel capacity depends on $L_{\min}$, contrary to the probabilistic channel, where it depends on the average substring length. More precisely, assume k information strings, each of length n, are transmitted through the channel (where for the single-strand torn-paper channel $k = 1$) and further assume that $\log(k) = o(n)$. We showed that if

$$L_{\min} = (a + o_{nk}(1)) \cdot \log(nk),$$

for some $a \geq 1$, then the rate of any $(L_{\min}, L_{\max})$-multistrand torn-paper code is at most $1 - \frac{1}{a} + o_{nk}(1)$.

To address the challenge of overcoming segmentation noise, we suggested and analyzed several code constructions for the torn-paper channel. Initially, we examined the code construction for the probabilistic torn-paper channel, as described in [66], and showed that its rate doesn't match the latter upper bound, i.e., we showed that this construction is suboptimal in adversarial scenarios. Subsequently, we developed efficient constructions encompassing several extensions of the single-strand torn-paper channel, including multistrand storage, substitution errors, or incomplete coverage. Importantly, our proposed constructions have linear runtime encoders and decoders, and the resulting codes achieve asymptotically optimal rates.

Building upon these results, in [84], we extended the model to also consider overlaps between two consecutive substrings. The new model assumes an information string is observed through an arbitrary collection of its substrings, where the length of each retrieved substring and the length of overlap between consecutive substrings are bounded from below. Following this model, Wei et al. [78] recently studied erroneous setups for both the single-strand and the multistrand scenarios.

While our analysis grapples with worst-case scenarios, the importance of considering a more realistic depiction involving an adversarial channel where the average segment lengths

are constrained is recognized. In such a scenario, adversaries can segment a non-negligible portion of the channel input into short substrings, thereby amplifying the challenge of coding for this channel. This complexity underscores the need for further exploration, leaving it as a compelling avenue for future research. An additional promising direction for future research is applying our methods to a generalized channel, including multiple sources of noise concurrently. The study of the torn-paper channel (with its variants and generalizations) under edit errors, including insertions/deletions in addition to substitutions, is also considered of great interest for applications to DNA data storage.

Continuing our exploration, in Chapter 6, we delved into the problem of almost-balanced sequences. This concept, which extends the well-known balanced Knuth code [37], addresses a fundamental challenge in constrained coding and carries significant implications for the reliability of DNA storage systems. Within the context of DNA storage, maintaining approximately balanced GC content (typically falling between 45% to 55% GC ratio) within stored sequences emerges as a powerful strategy for error mitigation.

The study of optimal almost-balanced sequences addressed the challenge of designing efficient algorithms with small redundancy for sequences with Hamming weight between $0.5n \pm \varepsilon(n)$. We presented a novel approach leveraging iterative methods and arithmetic coding to construct almost balanced codes with a single redundancy symbol and an average of $\mathcal{O}(n)$ operations, each requiring the multiplication of two rational numbers. Notably, our method achieved the optimal balanced order of $\varepsilon(n) = \Theta(\sqrt{n})$. Furthermore, we extended our method to non-binary, almost polarity-balanced sequences for even q and almost symbol-balanced sequences for $q = 4$, offering asymptotically optimal solutions for almost-balanced sequences in both binary and non-binary alphabets.

While our constructions achieve an optimally balanced order of $\Theta(\sqrt{n})$, there persists a multiplicative gap between theoretical bounds on $\varepsilon(n)$ and the actual values applicable in our constructions. For example, in the binary case, our suggested construction supports $\varepsilon(n) = \alpha\sqrt{n}$, for any $\alpha > \sqrt{\ln(2)} \approx 0.8326$, while theoretical bound for this case suggests that there exists a single redundancy bit construction for any $\alpha \geq 0.34$. Bridging this gap and exploring tighter existence bounds for the non-binary case represent critical avenues for future research. Future work should also focus on analyzing the number of symbols needed to maintain sufficient precision in the multiplication of rational numbers. This would enable a more accurate evaluation of our method's average time and memory complexities and facilitate a detailed comparison with the enumerative coding approach.

In a parallel work [6], we introduced a universal approach for parametric constrained coding, accompanied by a methodology for combining multiple parametric constraints. While this represents a significant step forward, future research avenues remain open for designing universal approaches that can seamlessly combine parametric and non-parametric constraints, as well as integrating this universal iterative approach with error-correcting capabilities.

In a broader context, the successful design of coding schemes tailored to the unique characteristics of DNA storage systems holds the promise of bringing us closer to practical implementation. Exploring the intricacies of such codes and their integration with each other presents a variety of intriguing challenges that warrant further investigation in future studies.

10.3 Towards Practical DNA Storage Systems

Part III of the thesis delves into the practical considerations and challenges of transitioning DNA storage from theory to application. This segment encompasses three distinct works, each contributing to different aspects of practical DNA storage systems.

In Chapter 7, we introduce a novel theoretical framework for representing information using DNA molecules, aiming to bypass the need for DNA synthesis, a slow and expensive process in traditional DNA storage methods. Inspired by concepts like DNA punch cards [69] and DNA composites [3, 14, 54], our approach explores DNA labeling tailored specifically for data storage purposes.

Enzymatic DNA labeling offers a versatile tool with applications across various scientific disciplines, including biochemistry, molecular biology, and biotechnology. Our contribution lies in formalizing a framework for DNA labeling optimized for data storage applications. This involves utilizing existing DNA molecules as templates for labeling, with the design of labels strategically chosen to represent encoded information. Our proposed approach draws on established chemical and biological protocols commonly used in medical and biological research, such as CRISPR-Cas9 and gRNA reagents for labeling. Through thorough exploration, we establish upper bounds on the maximal code size achievable under specific conditions and introduce an efficient encoder-decoder pair optimized for maximum code size.

To advance this novel approach, future research efforts should focus on adapting the model, bounds, and construction to a more realistic setup that accounts for noise. Essential to this progression is the identification of precise chemical and biological protocols capable of supporting the implementation of this approach, alongside the characterization and modeling of their corresponding noise mechanisms.

In our pursuit of practical DNA storage solutions, in Chapter 8, we presented a modular and comprehensive approach to DNA information retrieval. Our method seamlessly integrates DNN trained on simulated data, TP-based ECC, and a safety margin mechanism into a cohesive pipeline. By harnessing the parallel computational capabilities of DNNs and the efficiency of TP-based ECC, our approach aims to optimize scalability and accuracy in the information retrieval process. This modular architecture not only streamlines error correction but also enhances adaptability across various datasets and scenarios, laying a robust foundation for efficient and reliable DNA storage systems.

In our experimental evaluation, we applied our proposed approach to process 3.1MB of information using two different sequencing technologies. The results demonstrate significant improvements over existing solutions, showcasing up to a x3200 increase in speed, a 40% enhancement in accuracy, and achieving a code rate of 1.6 bits per base in a high noise regime. By tackling critical bottlenecks in the information retrieval process, our work presents a viable pathway for unlocking the full potential of DNA-based storage systems in practical applications.

Looking ahead, our research opens up several avenues for future exploration and development. Firstly, refining and optimizing our modular approach to further enhance scalability, accuracy, and efficiency, particularly in higher noise regimes, remains a priority. Additionally, extending our methodology to directly process the raw signals obtained from Nanopore sequencers represents an exciting direction. This endeavor will require the design of innovative

DNN architectures tailored to this specific problem domain as well as exploring new coding schemes tailored to address errors in raw signals, as opposed to the standard 4-ary DNA alphabet.

Furthermore, investigating novel integration approaches between ECC and DNN for applications beyond DNA storage holds significant potential. Exploring synergies between these techniques in diverse domains could lead to new avenues of research beyond the scope of DNA storage, broadening the impact of our work.

Lastly, Chapter 9, delves into strategies aimed at minimizing sequencing coverage depth in DNA storage systems to reduce sequencing costs and latency while ensuring retrieval with high probability. This research focuses on addressing three key problems: the MDS coverage depth problem, the coding coverage depth problem, and the coverage depth random access problem.

In the MDS coverage depth problem, it is assumed that the data is encoded using an $[n, k]$ MDS code. Through comprehensive analysis, we demonstrate that the coverage depth is minimized if and only if the channel has a uniform distribution. Additionally, we establish upper and lower bounds on the probability distribution of the required number of reads. The coding coverage depth problem extends our setup beyond MDS codes and considers the use of general codes. Under this setup, we demonstrated the superiority of MDS codes in minimizing the expected number of reads required for information retrieval.

By addressing the coverage depth random access problem, we aimed to improve data retrieval efficiency by minimizing the time required to retrieve individual data units from the storage system. Through meticulous analysis, we found that the expected number of reads which is needed for retrieval using the identity code, systematic MDS codes, and several other codes with dimension k is exactly k. Then, we presented and studied several innovative code constructions, which demonstrate that the coverage can be reduced below k. Lower bounds on the required number of reads are provided as well.

In a recent work [28], we extended the study of the coverage depth random access problem and offered additional understanding regarding several families of codes. Another extension to the random access setup was studied in [1], however, the goal is not to decode a single strand but a group of strands that constitute a single file. A related concept was also explored in [9], where the authors investigated the trade-offs between the reading costs, directly associated with coverage depth and the writing costs. The non-random access scenario of the DNA coverage depth problem was further extended in [53] to support the setup of the combinatorial composite of DNA shortmers [54].

Looking ahead, these results lead to several directions for future exploration and development. Extending our findings to diverse noise models and channel distributions is a promising avenue for further investigation. Additionally, in our work, the noisy channel was modeled by a parameter t, under the assumption that retrieval succeeds with probability 1 given t or more noisy copies and fails otherwise. A highly relevant extension to this noise model is to consider the more realistic behavior of the channel, in which the success probability can increase or decrease as a function of the number of noisy copies, i.e., as a function of the cluster size.

By tackling challenges related to sequencing costs, error correction, information redundancy, and synthesis costs, Part III contribute to addressing practical challenges and advancing the state-of-the-art in DNA storage systems. These research efforts offer valuable insights and solutions for building efficient, scalable, and commercially viable DNA-based data stor-

age systems. As the field continues to evolve, these pioneering contributions pave the way for realizing the full potential of DNA storage in real-world applications.

10.4 Summary

In conclusion, this dissertation significantly advances the field of DNA-based storage through a comprehensive exploration of theoretical insights, practical solutions, and innovative approaches tailored to address the unique challenges inherent in DNA storage systems. By laying a solid foundation, our research paves the way for the development of more efficient, reliable, and cost-effective DNA storage solutions, thereby bringing us closer to realizing the full potential of DNA as a storage medium for the Zettabyte era and beyond. However, our journey has also illuminated a diverse array of new challenges and opportunities, reminding us that the pursuit of knowledge is an ongoing expedition filled with endless possibilities. As eloquently captured in the following aphorism:

> *"The horizon of discovery expands with each step we take into the unknown".*

Bibliography

[1] H. Abraham, R. Gabrys, and E. Yaakobi, "Covering All Bases: The Next Inning in DNA Sequencing Efficiency," in *2024 IEEE International Symposium on Information Theory (ISIT)*, 2024.

[2] N. Alon and J. H. Spencer, *The Probabilistic Method,* John Wiley & Sons, 2016.

[3] L. Anavy, I. Vaknin, O.Atar, R. Amit and Z. Yakhini, "Data storage in DNA with fewer synthesis cycles using composite DNA letters". *Nature Biotechnology,* vol. 37, no. 10, pp. 1229–1236, 2019.

[4] G. B. Arfken, H. J. Weber, and F. E. Harris, "Chapter 13 - Gamma function," in *Mathematical Methods for Physicists (Seventh Edition)*, Academic Press, pp. 599–641, https://www.sciencedirect.com/topics/mathematics/digamma-function, 2013.

[5] J. L. Banal, et al., "Random access DNA memory using Boolean search in an archival file storage system," *Nature Material*, vol. 20, no. 9, pp. 1272–1280, 2021.

[6] D. Bar-Lev, A. Kobovich, O. Leitersdorf, and E. Yaakobi, "Universal framework for parametric constrained coding," *arXiv preprint arXiv:2304.01317*, 2023.

[7] D. Bar-Lev, I. Orr, O. Sabary, T. Etzion, and E. Yaakobi, "Deep DNA storage: Scalable and robust DNA storage via coding theory and deep learning," *arXiv preprint arXiv:2109.00031*, 2021.

[8] S. R. Blackburn, "Non-overlapping codes," *IEEE Transactions on Information Theory*, vol. 61, no. 9, pp. 4890–4894, 2015.

[9] S. Chandak et al., "Improved read/write cost tradeoff in DNA-based data storage using LDPC codes," *Annual Allerton Conference on Communication, Control, and Computing*, 2019.

[10] I. Chatzigeorgiou, "Bounds on the Lambert Function and their application to the outage analysis of user cooperation," *IEEE Communications Letters*, vol. 17, no. 8, pp. 1505–1508, 2013.

[11] G. Chaykin, O. Sabary, N. Furman, D. Ben-Shabat, and E. Yaakobi, "DNA-storalator: end-to-end DNA storage simulator," *13th Annual Non-Volatile Memories Workshop* 2022.

[12] B. Chen, W. Zou, H. Xu, Y. Liang, and B. Huang, "Efficient labeling and imaging of protein-coding genes in living cells using CRISPR-tag," *Nature communications*, vol. 9, no. 1, p. 5065, 2018.

[13] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations." *The Annals of Mathematical Statistics*, vol. 23, no. 4, pp. 493–507, 1952.

[14] Y. Choi et al., "High information capacity DNA-based data storage with augmented encoding characters using degenerate bases," *Scientific reports*, vol. 9, no. 1, p. 6582, 2019.

[15] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017 pp. 1251–1258.

[16] T. M. Cover, "Enumerative Source Encoding," *IEEE Transactions on Information Theory*, vol. 19, no. 1, pp. 73–77, 1973.

[17] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth, "On the LambertW function," *Advances in Computational Mathematics*, vol. 5, pp. 329–359, 1996.

[18] T. Coughlin, "175 Zettabytes by 2025," *Forbes*, https://www.forbes.com/sites/tomcoughlin/2018/11/27/175-zettabytes-by-2025/, 2018.

[19] I. Csiszar, "ℓ-divergence geometry of probability distributions and minimization problems," *The Annal of Probabilty*, vol. 3, no. 1, pp. 146–158, 1975.

[20] J. Deen, C. Vranken, V. Leen, R. K. Neely, K. P. F. Janssen, and J. Hofkens, "Methyltransferase-directed labeling of biomolecules and its applications," *Angewandte Chemie International Edition*, vol. 56, no. 19, pp. 5182–5200, 2017.

[21] DNA data storage alliance, "Preserving our digital legacy: An introduction to DNA data storage," a publication of *DNA Data Storage Aliance*, 2021.

[22] O. Elishco, R. Gabrys, E. Yaakobi, and M. Médard, "Repeat-free codes," *IEEE Transactions on Information Theory*, vol. 67, no. 9, pp. 5749–5764, 2021.

[23] P. Erdős, and A. Rényi, "On a classical problem of probability theory," *Magyar Tud. Akad. Mat. Kutató Int. Közl* vol. 6, no. 1, pp. 215–220, 1961.

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

[25] W. Feller, *An introduction to probability theory and its applications*, John Wiley & Sons, vol. 1, 2nd edition, 1967.

[26] R. Feynman, "There's plenty of room at the bottom," *Engineering and Science, California Institute of Technology*, vol. 23, no. 36, pp. 22, 1960.

[27] P. Flajolet, D. Gardy, and L. Thimonier, "Birthday paradox, coupon collectors, caching algorithms and self-organizing search," *Discrete Applied Mathematics*, vol. 39, no. 3, pp. 207–229, 1992.

[28] A. Gruica, D. Bar-Lev, A. Ravagnani, and E. Yaakobi, "Reducing coverage depth in DNA storage: A combinatorial perspective on random access efficiency," *IEEE International Symposium on Information Theory (ISIT)*, 2024.

[29] D. Hanania, D. Bar-Lev, Y. Nogin, Y. Shechtman, and E. Yaakobi, "On the capacity of DNA labeling," in *2023 IEEE International Symposium on Information Theory (ISIT)*, 2023, pp. 567–572.

[30] L. He and M. Ye, "The size of Levenshtein ball with radius 2: Expectation and concentration bound," in *2023 IEEE International Symposium on Information Theory (ISIT)*, 2023, pp. 850–855.

[31] D. S. Hirschberg, "Bounds on the number of string subsequences," *Annual Symposium on Combinatorial Pattern Matching*, 1999, pp. 115–122.

[32] D. S. Hirschberg and M. Regnier, "Tight bounds on the number of string subsequences," *Journal of Discrete Algorithms*, vol. 1, no. 1, pp. 123–132, 2000.

[33] K. A. S, Immink and J. H. Weber, "Very efficient balanced codes," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 188–192, 2010.

[34] K. A. S. Immink, J. H. Weber, and H. C. Ferreira, "Balanced runlength limited codes using Knuth's algorithm," in *2011 IEEE International Symposium on Information Theory (ISIT)*, 2011, pp. 317–320.

[35] S. Y. Itoga, "The string merging problem," *BIT Numerical Mathematics*, vol. 21, no. 1, pp. 20–30, 1981.

[36] J. Jeffet, S. Margalit, Y. Michaeli, and Y. Ebenstein, "Single-molecule optical genome mapping in nanochannels: Multidisciplinarity at the nanoscale," *Essays in Biochemistry*, vol. 65, no. 1, pp. 51–66, 2021.

[37] D. Knuth, "Efficient balanced codes," *IEEE Transactions on Information Theory*, vol. 32, no. 1, pp. 51–53, 1986.

[38] A. Kobovich, O. Leitersdorf, D. Bar-Lev, and E. Yaakobi, "Codes for constrained periodicity," in *2022 IEEE International Symposium on Information Theory and its Applications (ISITA)*, 2022.

[39] E. M. LeProust et al., "Synthesis of high-quality libraries of long (150mer) oligonucleotides by a novel depurination controlled process," *Nucleic Acids Research*, vol. 38, no. 8, pp. 2522–2540, 2010.

[40] M. Levy and E. Yaakobi, "Mutually uncorrelated codes for DNA storage," *IIEEE Transactions on Information Theory*, vol. 65, no. 6, pp. 3671–3691, 2019.

[41] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics. Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[42] V. I. Levenshtein, "Maximum number of words in codes without overlaps," *Problemy Peredachi Informatsii*, vol. 6, no. 4, pp. 88–90, 1970.

[43] V. I. Levenshtein, "Efficient reconstruction of sequences from their subsequences or supersequences," *Journal of Combinatorial Theory. Series A*, vol. 93, no. 2, pp. 310–332, 2001.

[44] Y. Liron and M. Langberg, "A characterization of the number of subsequences obtained via the deletion channel," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2300–2312, May 2015.

[45] H. Ma, A. Naseri, P. Reyes-Gutierrez, S. A. Wolfe, S. Zhang, and T. Pederson, "Multicolor CRISPR labeling of chromosomal loci in human cells," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 112, no. 10, pp. 3002–3007, 2015.

[46] F. J. MacWilliams and N. J. A. Sloane, "The theory of error-correcting codes," *Elsevie*, vol. 16, 1997.

[47] B. H. Marcus, R. M. Roth, and P. H. Siegel, "An introduction to coding for constrained systems," San Diego, CA, USA:Lecture Notes, 2001. [Online]. Available: `http://www.math.ubc.ca/~marcus/Handbook/`

[48] D. Markowitz, "Biology is all you need," *The International Conference for High Performance Computing, Networking, Storage, and Analysis*, 2023.

[49] R. Mascella and L. G. Tallini, "On symbol permutation invariant balanced codes," in *2005 IEEE International Symposium on Information Theory (ISIT)*, 2005, pp. 2100–2104.

[50] R. Mascella and L. G. Tallini, "Efficient m-ary balanced codes which are invariant under symbol permutation," *IEEE Transactions on Computers*, vol. 55, no. 8, pp. 929–946, 2006.

[51] M. Mitzenmacher, "A survey of results for deletion channels and related synchronization channels," *Probability Surveys* vol. 6, pp. 1–33, 2009.

[52] A. Moter and U. B. Göbel, "Fluorescence in situ hybridization (FISH) for direct visualization of microorganisms," *Journal of Microbiological Methods*, vol. 41, no. 2, pp. 85–112, 2000.

[53] I. Preuss, B. Galili, Z. Yakhini, and L. Anavy, "Sequencing coverage analysis for combinatorial DNA-based storage systems," *IEEE Transactions on Molecular, Biological, and Multi-scale Communications*, 2024.

[54] I. Preuss, M. Rosenberg, Z. Yakhini, and L. Anavy, "Efficient DNA-based data storage using shortmer combinatorial encoding," *Scientific Reports*, vol. 14, no. 1, p. 7731, 2024.

[55] S. Nassirpour, I. Shomorony, and A. Vahid, "Reassembly codes for the chop-and-shuffle channel, " *arXiv preprint arXiv:2201.03590*, 2022.

[56] D. J. Newman, "The double dixie cup problem," *The American Mathematical Monthly*, vol. 67, no. 1, pp. 58-61, 1960.

[57] T. T. Nguyen, K. Cai, and K. A. S. Immink, "Binary subblock energy-constrained codes: Knuth's balancing and sequence replacement techniques," in *2020 IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 37–41.

[58] L. Organick et al., "Random access in large-scale DNA data storage," *Nature Biotechnology*, vol. 36, no. 3, pp. 242–248, 2018.

[59] A. N. Ravi, A. Vahid, and I. Shomorony, "Capacity of the torn paper channel with lost pieces," in *2021 IEEE International Symposium on Information Theory (ISIT)*, 2021, pp. 1937–1942.

[60] J. Rissanen and G. G. Langdon, "Arithmetic coding," *IBM Journal of Research and Development*, vol. 23, no. 2, pp. 149–162, 1979.

[61] B. Ryabko, "A general method for the development of constrained codes," *arXiv preprint arXiv:2405.14570*, 2024.

[62] O. Sabary, Y. Orlev, R. Shafir, L. Anavy, E. Yaakobi, and Z. Yakhini "SOLQC: Synthetic oligo library quality control tool," *Bioinformatics*, vol. 37, no. 5, pp. 720–722, 2021.

[63] F. Sala and L. Dolecek, "Counting sequences obtained from the synchronization channel," *2013 IEEE International Symposium on Information Theory (ISIT)*, 2013, pp. 2925–2929.

[64] E. Sandifer, "How Euler did it". *Washington: Mathematics Association of America*, 2007.

[65] C. Schoeny, A. Wachter-Zeh, R. Gabrys, and E. Yaakobi, "Codes correcting a burst of deletions or insertions," *IEEE Transactions on Information Theory*, vol. 63, no. 4, pp. 1971–1985, 2017.

[66] I. Shomorony and A. Vahid, "Torn-paper coding," *IEEE Transactions on Information Theory*, vol. 67, no. 12, pp. 7904–7913, 2021.

[67] J. Spencer, "Asymptotic lower bounds for Ramsey functions," *Discrete Mathematics*, vol. 20, pp. 69–76, 1977.

[68] T. G. Swart and J. H. Weber, "Efficient balancing of q-ary sequences with parallel decoding," *2009 IEEE International Symposium on Information Theory (ISIT)*, 2009, pp. 1564–1568.

[69] S. K. Tabatabaei et al., "DNA punch cards for storing data on native DNA sequences via enzymatic nicking," *Nature communications*, vol. 11, no. 1, p. 1742, 2020.

[70] L. G. Tallini and B. Bose, "Balanced codes with parallel encoding and decoding," *IEEE Transactions on Computers*, vol. 48, no. 8, pp. 794–814, 1999.

[71] L. G. Tallini, R. M. Capocelli, and B. Bose, "Design of some new efficient balanced codes," *IEEE Transactions on Information Theory*, vol. 42, no. 3, pp. 790–802, 1996.

[72] A. J. Van Wijngaarden and K. A. S. Immink, "On the construction of constrained codes employing sequence replacement techniques," in *1997 IEEE International Symposium on Information Theory (ISIT)*, 1997, p. 144.

[73] A. J. Van Wijngaarden and K. A. S. Immink, "Construction of maximum run-length limited codes using sequence replacement techniques," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 200–207, 2010.

[74] A. Vaswani, et al. "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[75] G. Wang and Q. Wang, "On the size distribution of the fixed-length Levenshtein balls with radius one," *Designs, Codes and Cryptography*, pp. 1–13, 2024.

[76] J. H. Weber and K. A. S. Immink, "Knuth's balanced codes revisited," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1673–1679, 2010.

[77] J. H. Weber, K. A. S. Immink, P. H. Siegel, and T. G. Swart, "Polarity-balanced codes," in *2013 Information Theory and Applications Workshop (ITA)*, 2013, pp. 1–5.

[78] H. Wei, M. Schwartz, and G. Ge, "Reconstruction from noisy substrings," *arXiv preprint arXiv:2312.04790*, 2023.

[79] J. Wolf, "On codes derivable from the tensor product of check matrices," *IEEE Transactions on Information Theory*, vol. 11, no. 2, pp. 281–284, 1965.

[80] T. Xiao, M. Singh, E. Mintun, T. Darrell, P. Dollár, R. B. Girshick, "Early convolutions help transformers see better," *Advances in Neural Information Processing Systems*, vol. 34, pp. 30392–30400, 2021.

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

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

[83] S. M. H. T. Yazdi, H. M. Kiah, E. Garcia-Ruiz, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: trends and methods," *IEEE Transactions on Molecular, Biological and Multi-Scale Communications* vol. 1, no. 3, pp. 230–248, Sep. 2015.

[84] Y. Yehezkeally, D. Bar-Lev, S. Marcovich, and E. Yaakobi, "Generalized unique reconstruction from substrings," *IEEE Transactions on Information Theory*, vol. 69, no. 9, pp. 5648–5659, 2023.

[85] Y. Yehezkeally and N. Polyanskii, "On codes for the noisy substring channel," *2021 IEEE International Symposium on Information Theory (ISIT)*, 2021, pp. 1700–1705.

[86] A. P. Young, D. J. Jackson, and R. C. Wyeth, "A technical review and guide to RNA fluorescence in situ hybridization," *PeerJ*, vol. 8, p. e8806, 2020.

בנוסף העבודה מציעה מגוון תוצאות הנוגעות לגדלים של אנטי-צופנים במטריקת ה-FLL, כמו גם גדלי החיתוכים של כדור הכנסה וכדור הסרה. מחקרים אלה שופכים אור על התנהגות השגיאות המאפיינות מערכות אחסון מבוססות דנ"א, ומציעים תובנות לגבי הגבולות והיכולות של מערכות אלו.

בנוסף, במטרה לקדם טכניקות קידוד הקשורות לאחסון נתונים מבוסס דנ"א, עבודה זו מציגה סכמות קידוד חדשות ויעילות להתמודדות עם אתגרים הייחודיים למערכות אחסון מבוססות דנ"א, כגון שבירה של מולקולות ה-דנ"א. יתרה מכך, העבודה מתעמקת בבעיית הרצפים הכמעט מאוזנים. בעיה זו נותנת מענה לאתגר מהותי בתחום של קודי אילוצים ובעלת השלכות משמעותיות על המהימנות של מערכות אחסון נתונים מבוססות דנ"א. באחסון נתונים מבוסס דנ"א שמירה על כמות מאוזנת של G ו-C יחד (בדרך כלל בין 45% ל-55%) ברצפים המאוחסנים הינה אחת השיטות הידועות להפחתת כמות השגיאות במערכת.

לבסוף, עבודה זו מגשרת בין תיאוריה לפרקטיקה, ומציעה מספר פתרונות לאתגרים המונעים כיום שימוש מסחרי במערכות אחסון נתונים מבוססות דנ"א. העבודה מציגה הוכחת-היתכנות חלוצית לאחסון ואחזור נתונים ממערכת אחסון מבוססת דנ"א באמצעות שיטה המשלבת רשתות עצביות מלאכותיות עם אסטרטגיות קידוד מתקדמות. בנוסף, כדי להפחית את עלויות וזמני הריצוף (תהליך קריאת המידע ממערכת אחסון מבוססת דנ"א) תוך שמירה על הסתברות גבוהה לאחזור המידע, מוצגת ונלמדת בעבודה זו בעיית הכיסוי בדנ"א. בעיה זו דנה באיזון העדין הכרוך בעלויות הריצוף, משך הריצוף ודיוק אחזור המידע. יתר על כן, עבודה זו מציגה מודל תיאורטי חדש לייצוג מידע באמצעות מולקולות דנ"א, במטרה להתגבר על הצורך בסנתוז דנ"א, תהליך איטי ויקר המשמש כיום לכתיבת המידע בשיטות אחסון מבוססות דנ"א.

לסיכום, מחקר זה מעשיר את מתודולוגיות העבודה באחסון נתונים מבוסס דנ"א על ידי התייחסות למאפייני השגיאה המובנים במערכות כנזכר לעיל והצעת סכימות קידוד חדשניות העונות לאתגרים הייחודיים שמציבות מערכות אלה. השילוב של רשתות עצביות מלאכותיות מדגים את האפשרות לסילומיות באחזור המידע. במקביל, מודלים ושאלות תיאורטיות חדשות מנותחים במטרה להפחית את הזמן והעלויות הקשורים לריצוף וסנתוז מולקולות דנ"א. עבודות אלו יחדיו מהוות צעד משמעותי לקראת פיתוח פתרונות מעשיים לאחסון נתונים מבוסס דנ"א.

תקציר

כמות הנתונים והמידע הדיגיטלי המיוצר ע"י האנושות הולכת וגדלה בקצב מעריכי מידי יום, ועל פי הערכות של המכון הבינלאומי לנתונים (International Data Corporation, IDC), כמות המידע שנצטרך לאחסון בשנת 2025 צפויה להיות גדולה מפי חמישה מכמות המידע שהיה קיים ב-2018 ולהגיע ל-180 זטה-בייט (180ZB). הגידול המשמעותי במידע תלוי במספר גורמים, ביניהם השימוש הגובר בטלפונים חכמים, רשתות חברתיות ושירותי הזרמת מדיה, צמיחה של טכנולוגיות בינה מלאכותית ועוד.
הטכנולוגיות הללו יוצרות מגוון עשיר של נתונים מסוגים שונים, כמו מידע אישי, תכנים ממדיות חברתיות, מידע עסקי, נתונים רפואיים, מידע מחיישנים וכל מידע אחר שמאוחסן במחשב.

הריבוי ההולך וגדל של מידע שאנו יוצרים וצריכים לאחסן מציב את האנושות בפני אתגרים רבים הקשורים לאחסון הנתונים הללו, כאשר העיקרי שבהם הוא שטכנולוגיות האחסון הקיימות היום לא יעמדו בעומס הביקוש לאחסון. לכך השלכות מרחיקות לכת, שכן העולם הטכנולוגי בו אנו חיים תלוי באחסון מוצלח של מידע דיגיטלי. אחד הפתרונות הבולטים לפער זה הוא אחסון נתונים מבוסס דנ"א (DNA), המאפשר אחסון של מידע דיגיטלי על גבי במולקולות דנ"א המיוצרות באופן מלאכותי. שיטה זו נהנית מיתרונות משמעותיים על פני פתרונות אחסון מגנטיים ואופטיים כגון צפיפות מידע יוצאת דופן, עמידות לזמן ארוך וצריכת חשמל זניחה לשמירה על תקינות ושלמות המידע.

מחקר דוקטורט זה מתעמק במאפיינים מהותיים של השגיאות הנפוצות במערכות אחסון מבוססות דנ"א, במטרה לפתח אסטרטגיות קידוד עמידות ואלגוריתמים חדשניים לשיפור מהימנות, יעילות, סילומיות וחסכון בעלויות. יחדיו, העבודות המוצגות במחקר זה מקדמות את ההיתכנות של אחסון נתונים מבוסס דנ"א תוך כדי תרומה לבעיות תאורטיות בתחום מחקר זה.

עבודת מחקר זו מנתחת מבנים קומבינטוריים הקשורים לשגיאות הנפוצות במערכות אחסון מבוססות דנ"א, כמו שגיאות הכנסה והסרה. בפרט, העבודה מציגה ניתוח מפורט של גדלי כדורי שגיאה תחת מטריקת ה-FLL, שהינה המטריקה שיש לקחת בחשבון כאשר מספר ההכנסות שווה למספר ההסרות.

מיוחדת לחתול שלי, סטיבן, על שהכניס שמחה לאין שיעור לחיי. מהישיבה לצידי במהלך אינספור פגישות זום ועד להשתתפות בכל תרגול והרצאה שהצגתי באופן מקוון, היית בן לוויה בלתי צפוי אך משמח לאורך המסע הזה.

אני מודה למרים ואהרון גוטווירט, הקרן הלאומית למדע והפקולטה למדעי המחשב ע"ש הנרי ומרילין טאוב בטכניון על התמיכה הכספית הנדיבה בהשתלמותי.

מודה לך על החברות בנינו שהייתה מקור לתמיכה והנאה רבה לאורך המסע הזה. העבודה עם שניכם הייתה זכות, ואני אסירת תודה על כל מה שהשגנו יחד.

ברצוני להודות גם לכל עמיתיי ושותפיי למחקר במהלך הדוקטורט. כפי שאמר פעם טרי פראצ'ט, "המחקר הטוב ביותר שאתה יכול לעשות הוא לדבר עם אנשים." כל אחד מכם תרם למסע שלי בצורה משמעותית. למדתי ונהניתי לעבוד עם כולכם. החוויה הזו לא הייתה אותו הדבר ללא ההתעניינות, התמיכה, שיתוף הפעולה והליווי שלכם. בין אם דרך עצות, הכוונה, או אפילו שיחות מזדמנות בארוחות צהריים והפסקות קפה - אני אסירת תודה על כל רגע שחלקנו, בין אם בזמנים של קושי ובין אם הצלחות משותפות. בפרט, ברצוני להודות לפרופ' זהר יכיני, ד"ר ראיין גבריס, פרופ׳ אורי רוטנשטרייך, ד"ר יונתן יחזקאלי, ד״ר אנינה גרויקה, ענבל פרויס, דגנית חנניה, אביטל בורוחובסקי, אוריאן לייטרסדורף ואדיר קובוביץ על תרומתם המשמעותית, התמיכה והחברות לאורך המסע הזה.

תודה מיוחדת לבוחני התזה שלי, פרופ' רוני רוט ופרופ' משה שוורץ, על הקריאה היסודית של עבודתי ועל הערותיהם המעמיקות. המשוב שלכם תרם רבות לשיפור האיכות של עבודת התזה הזו, ואני באמת מעריכה את הזמן והמאמץ שהושקעו מצידכם. יתרה מכך, ברצוני להביע את תודתי על נוכחותכם לאורך המסע הזה. לרוני, תודה על שתמיד השתתפת בסמינרים וההרצאות שלי, גילית עניין אמיתי במחקר שלי וחלקת רעיונות מעולים והצעות מועילות. אתגרת אותי ללא הרף ותרמת לחידוד עבודתי ובישוריי ולצמיחתי בחוקרת ומחנכת. למשה, אני אסירת תודה על כך שתמיד הייתי זמין לייעץ ולהכווין והצעת אוזן קשבת, במיוחד לגבי שאיפותיי האקדמיות.

אני גם רוצה להביע את הערכתי לצוות המחלקה למדעי המחשב בטכניון על תמיכתם המתמדת. בפרט, ברצוני להודות ליפעת חן-סולומון על שסייעה לי בכל צורך מנהלי במהלך הלימודים.

לבסוף, ברצוני להקדיש את התזה למשפחתי. לבעלי, שחל ויסלדר, שהיה לצידי לאורך כל המסע הזה - תודה על התמיכה הבלתי מעורערת, העידוד והעניין האמיתי והבנה שלך במחקר שלי ובעבודתי. הסבלנות וההבנה שלך נתנו לי את המרחב והזמן לעבוד, והחוסן שלך הפך לבסיס שעליו יכולתי להישען ולבנות את שלי. אתה היית, וממשיך להיות, מקור הכוח והנחישות שלי. להורי, לאה ואלברט בר-לב, לאחותי עדן בר-לב, ולסבי וסבתי, תמרה ויעקב אליגולשווילי ולסבתי זיוה דניאל - תודה שגידלתם אותי והקניתם לי את הכלים, הסקרנות והבישורים הרבים שאפשרו לי לצלוח את המסע הזה. תמיד עודדתם את האופי הסקרן שלי וסיפקתם לי את כל מה שהייתי צריכה כדי להצליח. חשוב מכך, האהבה, האכפתיות והתמיכה האינסופיים שלכם, דרך כל אתגר שהחיים הביאו, עיצבו אותי לאדם שאני היום. האמונה שלכם בי, כבר מגיל צעיר, הייתה מקור מתמיד למוטיבציה. אני אסירת תודה לנצח על נוכחותכם בחיי. אחרון חביב, תודה

המחקר בוצע בהנחייתם של פרופ׳ טובי עציון ופרופ' איתן יעקובי, בפקולטה למדעי המחשב בטכניון.

תודות

בראש ובראשונה ברצוני להביע את תודתי העמוקה למנחים שלי, פרופ' טובי עציון ופרופ' איתן יעקובי. הבחירה לעבוד עם שניכם הייתה ללא ספק אחת ההחלטות הטובות ביותר שעשיתי בחיי ואני אסירת תודה על ההזדמנות ללמוד ולצמוח תחת הנחייתכם והדרכתכם.

לטובי, תודה על העידוד התמידי ועל כך שלימדת אותי מה המשמעות של להיות חוקרת טובה. למדתי ממך הרבה מעבר לכלים מחקריים, ובפרט, למדתי כיצד להתנהל בעולם האקדמי. האמנת בי גם בזמנים בהם פקפקתי בעצמי, והביטחון שהקנית בי שימש אותי רבות לאורך המסע הזה וימשיך לשמש אותי בעתיד. הסבלנות האינסופית שלך, הנכונות לעזור בכל עניין, והעצות המעולות שלך שתמיד ניתנו מתוך אכפתיות וחשיבה על טובתי - הן תכונות שאני מעריכה מאוד. איתן, הינך מודל לחיקוי, ותמיד נתת לי השראה לפרוץ גבולות. ההתלהבות והסקרנות האינטלקטואלית שלך הניעו אותי לחקור רעיונות חדשים, לא משנה עד כמה הם נראו לא שגרתיים. תודה שתמיד עודדת את סקרנותי ותמכת ברעיונות שלי, גם אלו שנשמעו נועזים ביותר. שניכם פתחתם לי אינספור דלתות וסיפקתם הזדמנויות שרק יכולתי לחלום עליהן. תמיכתכם הבלתי מעורערת, החל מהשלבים הראשונים של המסע האקדמי שלי ועד לעצמאות במחקר, עיצבה אותי להיות החוקרת שאני היום. אני אסירת תודה לנצח על המלגות, ההזדמנויות, והכי חשוב, הידע שהענקתם לי. אני אעשה כמיטב יכולתי ליישם את כל מה שלמדתי מכם, ואני מקווה שאצליח להעביר הלאה ולו רק חלק ממה שקיבלתי מכם בהמשך הקריירה האקדמית שלי.

אני מודה מקרב לב לעומר צברי. היית חלק מהותי מהמסע הזה וממה שהפך אותו למספק אקדמית, ובד בבד למהנה ומרגש. תמיד יכולתי לסמוך עליך, בתור קולגה מקצועית וכחבר קרוב. בין אם חגגנו הצלחות או חלקנו פחדים וחלומות, התמיכה שלך ושיתוף הפעולה שלנו בפרויקטים רבים הפכו את החוויה הזו למעשירה מאוד. לד"ר איתי אור, אני אסירת תודה על אינספור דיונים ושיתוף רעיונות. הניסיון והידע שלך חשפו אותי למושגים ונקודות מבט חדשות ונהניתי מכל רגע של למידה. מעבר לזה, אני

תאוריה ופרקטיקה של זכרונות דנ"א

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר

דוקטור לפילוסופיה

דניאלה בר-לב

הוגש לסנט הטכניון – מכון טכנולוגי לישראל

סיוון התשפ"ד חיפה יולי 2024

תאוריה ופרקטיקה של זכרונות דנ"א

דניאלה בר-לב