DNA-Based Data Storage Systems: A Review of Implementations and Code Constructions

Olgica Milenkovic, *Fellow, IEEE*, and Chao Pan

***Abstract*— This invited review paper has the aim to acquaint the communication theory community with the emerging topic of molecular data storage. The exposition includes an overview of basic concepts in synthetic and computational biology and a discussion of diverse approaches used to implement such systems. It also describes new problems in communication and coding theory, and discusses some relevant results pertaining to DNA sequence profiles, coded trace reconstruction, coding for DNA punchcard systems and coding for unique reconstruction.**

***Index Terms*— Coded trace reconstruction, coding for DNA profiles, DNA-based data storage, string reconstruction.**

I. Introduction and Motivation

DESPITE numerous advancements in traditional data recording techniques, the emergence of Big Data platforms and the growing concern for energy conservation have presented challenges for the storage community to develop new nonvolatile, durable storage media that can handle ultrahigh volumes of data.

The potential use of macromolecules for data storage was recognized as far back as the 1960s when Richard Feynman outlined his nanotechnology vision in the talk "There is plenty of room at the bottom" [33]. Among the various macromolecules that can potentially serve as storage media, DNA molecules hold particular promise due to their unique properties such as durability, ultra-large information density, ease of amplification, readout compatibility and ability to perform computing via simple hybridization reactions. Under proper environmental conditions, DNA can preserve its contents for thousands of years, as demonstrated by the recovery of DNA from 30,000 years old Neanderthal and 700,000 years old horse bones [99]. In addition, DNA offers extremely high storage capacities, with a single human cell containing DNA strands that encode 6.4 gigabits of information within a mass of only approximately 3 picograms. The technologies for DNA amplification and synthesis have also reached unprecedented levels of efficiency and accuracy [103], while DNA sequencing has been a standard procedure for nearly two decades. DNA has also been successfully used as a building block for small-scale self-assembly based computers [102].

Building on the progress of DNA synthesis and sequencing technologies, two laboratories described the first architectures for archival DNA-based storage in 2012 and 2013 [25], [43]. The first architecture achieved a density of 700 TB/gram, while the second approach improved the density to 2 PB/gram. The improved results of the second approach may be attributed in part to the use of basic coding schemes such as Huffman coding, runlength coding, single parity-check coding, and repetition coding. Subsequent works [45] extended the coding approach of the second architecture to account for missing information-bearing DNA fragments via Reed-Solomon codes [95].

Further milestones in DNA-based data storage were reached through several innovations. The first innovation was the introduction of random-access and rewriting platforms enabled by controlled polymerase chain reaction (PCR) and/or overlap-extension PCR reactions [125]. The design of DNA PCR primers (addresses) from a coding-theoretic perspective, which initiated with [127], also played a crucial role in scaling up this approach for larger file sizes. The second innovation was the design of portable DNA-based data storage platforms that utilize long readout sequences and are accompanied by specialized pilot sequencing, multiple sequencing alignment, and constrained homopolymer (i.e., runlengths of the same symbol) coding approaches [124]. This development has given rise to new challenges such as *coded trace reconstruction* [24] and various forms of synchronization error-correction. The third milestone involved an expansion of the molecular alphabet to include modified DNA bases [114] that can be read using commercially available nanopore devices coupled with deep learning solutions for base classification. Simultaneously, theoretical models for DNA "storage channels" have been proposed to rigorously analyze the above-described architectures [58], involving overlapping DNA fragments akin to those used in the original storage architectures of [25] and [43], and nonoverlapping information-bearing blocks which model pools without address sequences (see [49], [63] and references therein). These models have been the basis for further research on the fundamental aspects and capacity of DNA storage channels.

Despite this early success in developing DNA-based data storage systems, many issues remain unresolved, with the most important one being the high error rates resulting

Manuscript received 6 October 2023; revised 10 January 2024; accepted 9 February 2024. Date of publication 20 February 2024; date of current version 19 July 2024. This work was supported by the NSF Grant 2008125. The associate editor coordinating the review of this article and approving it for publication was E. Rosnes. *(Corresponding author: Olgica Milenkovic.)*

The authors are with the Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign, Urbana, IL 61801 USA (e-mail: milenkov@illinois.edu; chaopan2@illinois.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCOMM.2024.3367748.

Digital Object Identifier 10.1109/TCOMM.2024.3367748

from associated synthesis, access, and readout processes [34], [77], [96] and the extremely high cost of DNA synthesis. Additionally, known designs still lack the computational capabilities required to support operations on data stored in molecular media.

Sequencing errors have largely been mitigated through the use of existing [59], [67], [68], [117] and the design of specialized coding approaches [36], [45], [58], [62], [79]. These approaches can handle missing DNA fragments, in-fragment sequencing errors, and asymmetric errors caused by the specific molecular topologies of DNA bases. Handling other types of errors in molecular storage has been made possible through a combination of constrained coding, which avoids DNA patterns prone to synthesis or sequencing errors, prefix synchronized coding, which allows accurate access to blocks of DNA without disturbing other blocks in the DNA pool, and low-density parity-check (LDPC) codes [42], which provide redundancy for combating classical substitution errors.

A satisfactory solution to the problem of high-cost synthesis is still missing since synthesis is a sequential process that can currently only be made faster through parallel synthesis of shorter sequence blocks and subsequent ligation/attachment (an approach used by *Catalog*, www.catalogdna.com). The *DNA Punchcard* paradigm was introduced in [115] as a partial solution to the problem of sequential DNA synthesis and for the purpose of joint storage and computing. In this paradigm, native DNA (i.e., DNA retrieved from common bacterial species such as *E. coli*) is used as the storage media. Binary or ternary information is imprinted on the DNA by creating controlled nicks (i.e., holes) at specific locations of the molecular backbone. Since the sequence content of the DNA strings is known, retrieving the stored information is straightforward and close to error-free.

Although the DNA Punchcard system experiences a moderate density loss compared to traditional sequence encodings, it offers highly efficient parallel writing and unique massively parallel in-memory computing features [22], [122]. The in-memory computing model, known as SIMDNA, relies on carefully shifting and recreating nicks in multiple information-bearing DNA registers using specialized *strand displacement* reactions and combinatorial design rules. SIMDNA's most appealing feature is its ability to use the same instruction DNA strands to update all registers, regardless of their content.

Additionally, nicks can be overlaid on DNA strands that carry information to include rewritable data, such as metadata. This rewriting process involves sealing the nicks using native ligases [89] and then repunching the helix. The Punchcard method and its recent extension, known as DNA Typewriters [80], which operate *in vivo*, present new challenges related to constrained coding and error correction due to their unique information storage approach. In Punchcard systems, for instance, it is possible to choose nick locations that have significant differences in sequence content to avoid errors during punching. Nonetheless, the placement of nicks must still satisfy certain requirements regarding their distribution on the two DNA strands. Essentially, the placement of nicks should ensure the overall stability of the double-helix. These constraints lead to a new coding paradigm for sets and introduce intriguing questions related to *set discrepancy* analysis [35].

The aim of this overview article is to provide an accessible introduction to the key components of the previously described DNA-based storage systems. These components include DNA synthesis, PCR protocols for random access, synthetic biology concepts like gene editing using CRISPR complexes, sequencing techniques such as shotgun, nanopore (e.g., Oxford Nanotechnologies (ONT)) or Pacific Biosciences (PacBio) sequencing, as well as strand displacement molecular computation paradigms.

Additionally, the article will describe the fundamental concepts behind current DNA-based data storage architectures. It will explain how real biological challenges have influenced the design of coding solutions that are necessary to ensure reliable scaling and operation of these systems. Moreover, it will highlight the importance of expertise in coding theory to inspire new system designs and tackle practical challenges in system implementation. Special attention will be placed on reviewing the recent contribution to the field made by the author and her collaborators. For a summary of relevant concepts and terminology in synthetic and molecular biology, the interested reader is also referred to the earlier review [126] and the more recent monograph [105].

The manuscript is organized as follows. Section II contains a a review of basic properties of DNA molecules and a gentle introduction to relevant concepts from synthetic biology. Section III describes a collection of conceptually different approaches to DNA-based data storage system design and provides a short review of DNA strand displacement computational paradigms. Section IV presents a review of coding-theoretic results that were developed to deal with reliability and implementation issues encountered in DNA-based data storage systems. Selected open problems in coding theory are described in Section V.

II. Synthetic Biology Preliminaries

Deoxyribonucleic acid (DNA) is a macromolecule – a molecule made up of a large number of atoms. It is found in single-cell organisms (e.g., bacteria and viruses) as well as in the *mitochondria and cell nucleus* of higher organisms (eukaryotes), where the latter is a compartment within the cell of width $5-10\mu\text{m}$.

In eukaryotes, DNA takes the form of a right-handed double-helix. It consists of two periodic linear molecules that twist around each other, forming the *sugar-phosphate backbone* (see Figure 1). The sugar-phosphate backbone has a deterministic structure, alternating between a deoxyribose sugar molecule and a phosphate group. It does not carry useful information. Useful information is contained in the "space" between the linear molecules, where four different molecular structures, called *bases*, bind together in pairs through hydrogen bonds. The bases are adenine (A), guanine (G), cytosine (C), and thymine (T). Bases A and G, which have two carbon rings, are *purine bases* (depicted in Figure 1 by a hexagon-pentagon structure), while bases C and T, which have one carbon ring, are *pyrimidine bases* (depicted in

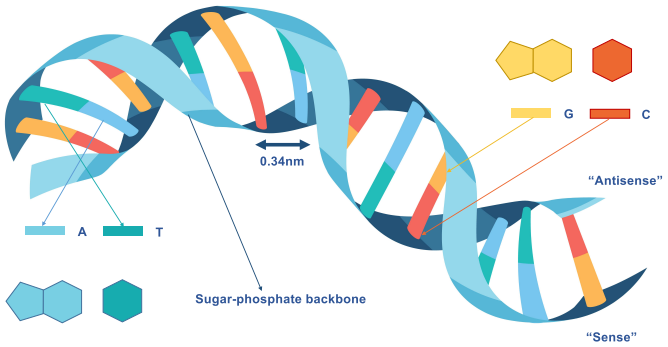


Fig. 1. Structure of the DNA macromolecule. The spacing between pairs of bases is a fraction of a nanometer (nm), andthis dense packing is responsible for the large storage capacity of the molecule.

Figure 1 by a hexagon). A molecular unit comprising one base, one sugar and one phosphate group is termed a *nucleotide*, and often used interchangeably with the term base.

There are two important observations to make about DNA bases. First, not all pairings are possible. According to the *Watson-Crick rule,* A only binds with T through two hydrogen bonds, and vice versa. Similarly, G only binds with C through three hydrogen bonds, and vice versa. While there are some rare exceptions, the Watson-Crick rule is generally considered a fundamental constraint for DNA molecules. As a result, the information-bearing sequence attached to one linear molecule is the (Watson-Crick) complement of the information-bearing sequence attached to the other linear molecule. For example, the complement of ATTCG is TAAGC.

Second, since bases are asymmetric molecules, we can orient DNA strings based on the numbering of the terminal carbon atom at the end of the string. Only the 3rd and 5th carbon can appear at the terminus of the deoxyribose sugar ring, and a string can be read from either the $3'$ carbon end or the $5'$ carbon end. The symbol $'$ is used to denote the carbon atoms in the sugar and is part of a standard chemistry vocabulary. For example, the string ATTCG used in the previous example may be read from the $3'$ to $5'$ end, written as $3'-\text{ATTCG}-5'$. This string is different from its reversal, which is written as $5'-\text{ATTCG}-3'$. If both base strings of a DNA molecule are read in the same direction, they represent reverse Watson-Crick complements. For the running example, if both strings are read from the $3'$ to $5'$ end, they would equal ATTCG and CGAAT. Alternatively, they can be written as $3'-\text{ATTCG}-5'$ and $5'-\text{TAAGC}-3'$. The strand running in the $5'-3'$ direction will be referred to as the *sense strand,* while the string running in the $3'-5'$ direction will be referred to as the *antisense strand.* These terms are borrowed from genetics and are based on the reading directions of protein-coding genes. Here, they are only used to refer to the orientation of the strands since no genes are involved.

The process of binding a sense and antisense strand to form a double-helix is called *hybridization,* and the process of separating the sense and antisense strand is called *denaturation.* Denaturation is typically achieved by heating up the DNA since thermal energy breaks down the hydrogen bonds and leads to the disassociation of the strands.

While not immediately evident, the aforementioned properties of DNA molecules are of significant importance in the context of DNA-based data storage system implementations.

For instance, consider the two purine bases, A and G, which each possess two carbon rings, resulting in a more similar chemical structure when compared to pyramidines. This similarity implies a higher likelihood of confusing them during sequencing (in contrast to, say, A and T). This observation also extends to the pyramidine bases. To address this issue of higher confusability between the pairs of pyrimidine and purine bases and the relatively lower confusability between purines and pyramidines, specialized data encoding protocols have to be used (as suggested for *asymmetric Lee distance codes* described in [36]).

Furthermore, the disparity in the number of hydrogen bonds formed between A and T versus G and C in the Watson-Crick pairings underscores the necessity of maintaining what is known as "balanced GC content" in information-bearing DNA strings. A small number of GC pairs may lead to instability in the DNA duplex while a large number may hinder efficient DNA synthesis and denaturation, as elaborated in the following section.

The significance of reverse Watson-Crick strings is evident in DNA replication, a process that involves creating two copies of DNA from a single template. During replication, the double helix gradually unravels, allowing each constituent string to serve as a template for generating a complementary strand. Outside the cell, replication is performed through a process called Polymerase Chain Reaction (PCR), which is also employed in the testing of viral diseases like Covid-19 (see Figure 2) and plays a crucial role in the unique approach to random access in DNA-based data storage [125].

DNA replication cannot commence without a specialized class of molecules known as "primers." Primers are short DNA fragments, roughly 20 bases in length, which are single-stranded. Primers facilitate the binding of enzymes (functional proteins) essential for replication of the DNA strands. To enable DNA content amplification and, consequently, random access, primers must adhere to several constraints. First, their "melting temperature," defined as the temperature at which 50% of the DNA in a solution exists in a double-stranded form and 50% in a single-stranded form, must closely match the range of temperatures $55-70^{\circ}\text{C}$. Maintaining an

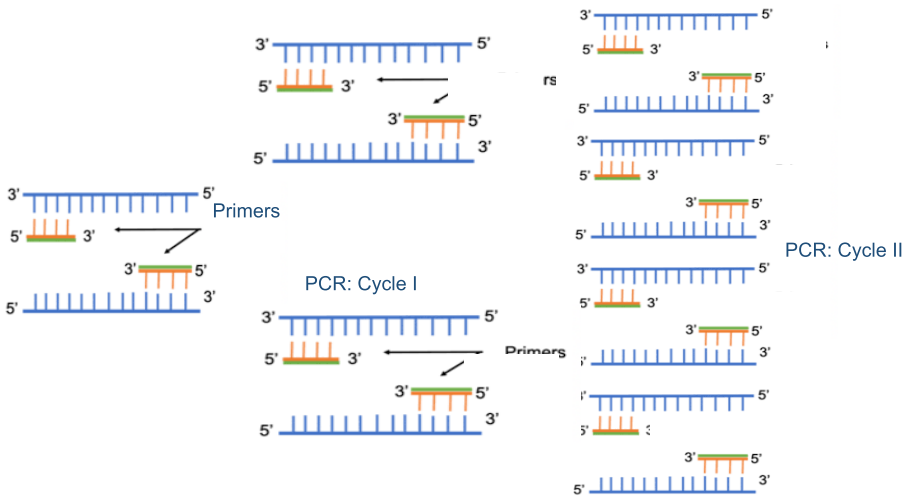


Fig. 2. Illustration of primer binding on denatured DNA. Two primers are required, one for the sense strand and another for the antisense strand. In each cycle of PCR amplification, utilizing Watson-Crick complementarity, two identical copies of the same strand are generated, ideally resulting in exponential growth in the concentration of the DNA product. The first random access protocol, described in [125], relies on the use of primers to amplify only a desired collection of DNA sequences. After performing PCR amplification on these DNA sequences for a sufficient number of cycles, there is an overwhelming probability that only the intended sequences are present in the pool. For information on microfluidic and self-rolled membrane random access systems that do not require PCR amplification, the interested reader is referred to [57].

appropriate melting temperature and binding stability hinges on a constant, balanced GC content. Numerous online platforms, such as *http://insilico.ehu.es/tm.php?formula=basic*, can be used to estimate melting temperatures of DNA strands.

Second, it is crucial to meet the "no-folding" and "no primer-dimer" constraints. Folding refers to the formation of a partially hybridized molecule through pairings of complementary bases on the same strand. Primer-dimer constraints, on the other hand, prevent two distinct single-stranded primers from hybridizing or partially hybridizing with each other. For a comprehensive exploration of the coding challenges associated with primer design constraints, interested readers are referred to [127].

The linear distance between two adjacent bases on DNA strings is approximately 0.34 nanometers, which implies that DNA can store 2 bits within this length. Consequently, the linear storage density of DNA is approximately 6×10^9 bits/m. More commonly, storage density is expressed in terms of bit-mass density, taking into account that the average mass of a nucleotide is 330 Daltons, with one Dalton equal to 1.66×10^{-24} grams. This translates to the ability to store 2 bits in a mass of 5.48×10^{-22} grams, or 3.6×10^{21} bits/gram. It is important to note that this represents the *physical storage density*, which is typically higher than the information storage density due to latter taking into account overheads for address, error-correction, and constraint coding. The reported information densities currently surpass by orders of magnitude those achievable by any other existing storage technology.

DNA can maintain its integrity for tens to hundreds of thousands of years when stored in a low-humidity, radiation-free environment. Given the ongoing drive for performance enhancements and cost reductions in DNA writing (synthesis) and reading (sequencing) technologies, especially in the fields of medical and fundamental molecular biology research, molecular storage platforms hold a unique promise among their competitors regarding future system compatibility.

A. DNA Synthesis and Sequencing: Building a DNA-Based Data Storage System

Building a basic DNA-based data storage system is indeed feasible, but it entails several essential components: sufficiently large financial resources, reliable synthetic DNA suppliers, and access to sequencing platforms such as Illumina or third-generation alternatives including ONT and PacBio. Such sequencers are readily used in genomic research laboratories, but with the exception of ONT systems, they are too expensive and bulky to be part of commercial readout systems.

The availability of sufficient funds is paramount, given that the DNA synthesis process is costly. This financial requirement stands as the primary impediment to the widespread adoption of molecular storage systems at scale. In the ensuing discussion, we elucidate the principles underpinning DNA synthesis and sequencing while also shedding light on potential errors that can arise during these intricate processes.

1) Synthesis: To synthesize user information into DNA, the first step involves introducing controlled redundancy into the original binary data string. This redundancy serves two purposes: facilitating various functionalities (including random access and content replication) and ensuring robustness (address redundancy is discussed in [126], while error-control coding is discussed in the sections to follow). Subsequently, the original binary information string is transformed (mapped) into a string over the DNA alphabet comprising four letters, A, T, G, C. Advancements in chemically modified DNA-based data storage have also paved the way for conversions into larger molecular alphabets, ranging from 8 to 11 letters [114].

In the next step, the quaternary data string is segmented into either overlapping or nonoverlapping substrings. These

digital substrings are converted into actual DNA strings harboring identical content. While overlapping substrings were initially employed in the early prototypes of DNA-based data storage [25], [43], they have been mostly abandoned due to their high coding redundancy and inefficiency of random access.

When synthesizing the DNA content, it is important to consider two factors: the lengths of the substrings and the format in which they are delivered, which is constrained by the synthesis technology used. For instance, when procuring products from *Integrated DNA Technologies (IDT),* customers can opt for what are referred to as *gBlocks.* gBlocks are double-stranded DNA strings with lengths of approximately up to $3,000$ base pairs (bps). They are primarily used for gene construction and play a pivotal role in genome editing (see *https://www.idtdna.com/pages/products/genes-and-gene-fragments/double-stranded-dna-fragments/*). Each gBlock is provided as an individual string, and users have the flexibility to choose the molar concentration of the product. Typically, gBlocks require the inclusion of prefix and suffix primer sequences to enable subsequent amplification of the relatively small volume of purchased synthetic DNA. The same primers are used in the PCR-based random access protocol. The advantages of gBlocks include their long length, which ensures a smaller proportion of the content dedicated to primer substrings, stable double-helix structure, as well as their ease of reading via ONT and PacBio devices. Additionally, each fragment is provided in a separate storage tube or well. However, it is important to note that gBlocks are associated with a higher synthesis cost per nucleotide when compared to their shorter single-stranded counterparts, described next.

As an alternative, one can opt for "DNA oligo pools" (*https://www.idtdna.com/pages/products/custom-dna-rna/*. These pools consist of unordered collections of numerous *short, single-stranded* DNA strings, referred to as *oligos*. For instance, *IDT oPools* are available in formats that encompass anywhere from 2 to 384 oligos per pool, with oligo lengths ranging from 4 to 350 nucleotides. It is guaranteed that each oligo is present at a concentration of 50 pmols. The most cost-effective package offers oPools with a per-base cost of approximately \$0.011. This cost, while significantly higher than that of traditional recording media, still represents a more budget-friendly alternative compared to that of gBlocks.

oPools come with their own set of advantages and disadvantages. Advantages of oPools include the previously mentioned cost-effectiveness and ease of handling. However, they also exhibit several drawbacks. Typically, oPools have a lower average synthesis fidelity, reduced stability, a propensity of oligos to hybridize with each other. Additionally, they are burdened by substantial primer overheads. Furthermore, if not synthesized to full lengths ranging from 150 to 300 bases, they cannot be directly read using third-generation sequencing devices. Detrimental for the underlying molecular storage systems is the problem of *missing oligos*, referring to the absence of one or more oligos requested for synthesis. Missing oligo errors arise due to many factors such as placement of the oligo to be synthesized on a microarray (or other type of) grid, their base content and others. It is also worth noting that the primers required for content amplification need to be purchased separately (https://www.idtdna.com/pages/products/custom-dna-rna/dna-oligos/custom-dna-oligos).

A simplified diagram of the steps used in commercial *phosphoramidite chemistry* DNA synthesis is depicted in 3. Specialized forms of all four types of nucleotides that can be attached to a growing DNA strand are kept in four separate repositories and retrieved according to the string being synthesized. The nucleotides contain special *protective groups*, depicted as triangles. When initially incorporated, the nucleotides' protective groups prohibit the attachment of other nucleotides, thereby ensuring that only one symbol is added at each incorporation time. Once the nucleotides of the current symbol are washes of, the protective groups are deactivated to allow for the incorporation of the next symbol.

To more precisely explain the sequencing process, assume next that the string ATTCGATGCC has already been synthesized and that we want to add the symbol A. In this case, we would flush the synthesis well of array containing the partially synthesized string with protected A nucleotides and the enzymes (including polymerases) necessary for synthesis. The protective group prevents unintentional incorporation of multiple DNA symbols in one round/cycle of synthesis. Specifically, it disables access to other nucleotides on the strand once it is added as part of the newly included nucleotide. After the nucleotide is incorporated, any unused A symbols need to be washed off to avoid contaminating the new pool of symbols (which may be different from A) in the next cycle of DNA string extension. Washing is not entirely precise, so some unused nucleotides from previous cycles may remain. However, due to extensive chemical error-correction of the strands, this imprecision does not result in a very likely error event (i.e., in practice, no errors involving repeated symbols are observed in gBlock DNA products, and only a small fraction of errors are typically observed in sequenced oPools, where the errors may have actually been introduced during sequencing). Nonetheless, in theory, simultaneous incorporation of multiple bases could lead to *sticky insertion errors* [81]. Once the washing process is completed, and in preparation for the next cycle of growth for the extended strand ATTCGATGCCA, the protective group is removed or deactivated using lasers light or other means. However, the deactivation process is also prone to errors, which may result in some strands being permanently "deactivated". In this case, one ends up with incompletely synthesized DNA oligos, which are usually removed by the vendor before delivering the product. In some cases, temporary deactivation or premature activation lead to oligos with burst deletions or insertions, respectively. Oligos with bursty deletions can be identified through their shorter length and removed. IDT products have a very low likelihood of containing synthesis errors of the aforementioned types, but the company may report synthesis issues related to unbalanced GC-content, *short repeats*, i.e., repeats of short DNA substrings, and *long homopolymers* (see also https://www.twistbioscience.com/faq/gene-synthesis/are-there-any-sequence-limitationsdesign-guidelines-genes-which-i-should-follow and [101]). The most significant errors are

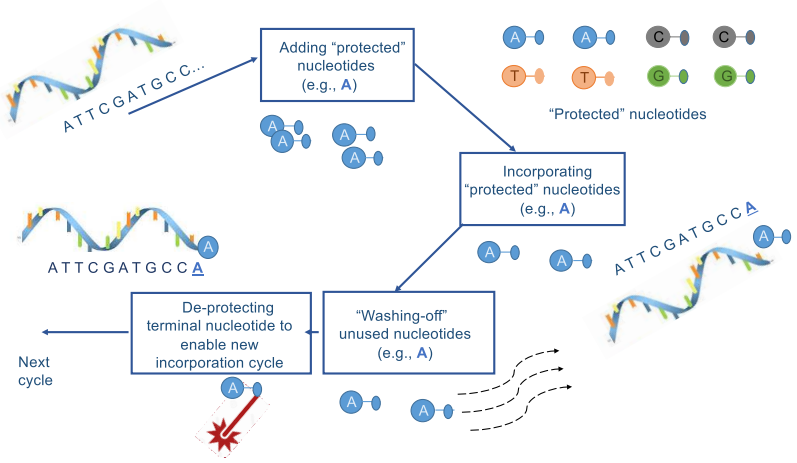


Fig. 3. High-level description of the synthesis process. One may think of using four types of different beads (nucleotides) which have to be stitched together sequentially to create the desired user content. The process is inherently sequential and involves multiple steps of nucleotide incorporation, washing and protective group deactivation that can lead to long synthesis latency (most providers report times of $1-10$ seconds/nucleotide).

typically missing or low-coverage oligos, where certain fragments were not synthesized to acceptable lengths or were synthesized inefficiently. These errors can be corrected using Reed-Solomon coding schemes described in [45] or through machine learning techniques used in 2DDNA systems [89]. Lastly, it is worth mentioning that the efficiency of synthesis schedules can be addressed analytically, as outlined in [53], [62] and [70].

2) Sequencing: There are currently many different technologies available for reading the content stored in DNA. One example is the sequencing-by-synthesis approach used by Illumina platforms, as shown on the left of Figure 4. Illumina devices, such as MiSeq, NovaSeq, and HiSeq, have a limitation in terms of the length of strands they can read, which is usually not more than 400 nucleotides. This technology is commonly used for reading pools of DNA oligos because the oligo lengths match the required sequencing lengths. The DNA fragments generated by Illumina and other sequencers are referred to as *reads* and are summarized in raw data files with the.fast or .fastq extension. The .fastq files not only contain read sequences but also information about quality scores of the symbols, allowing for assessment of the quality of the results. Illumina systems have high sequencing accuracy, although still not accurate enough for demanding storage applications (most systems currently operate with an error rate of less than $0.1-1\%$). Additionally, since multiple copies of the DNA strands are read simultaneously, consensus sequences can be easily formed by using majority counts for each position in the reads, as Illumina sequencing errors are mainly substitutions.

Third generation sequencers are capable of reading long gBlock data formats and are commonly known as *long read technologies.* One important long-read sequencing technology utilizes nanopores, which can provide single-molecule readouts of lengths ranging from $15,000$ to $20,000$ bases, or even longer. Nanopores are pores or holes embedded in membranes, with one or multiple pores on the same membrane. In the case of ONT nanopores, the pores are "biological" pores, such as proteins, and only double-stranded DNA is used for sequencing in order to control the speed at which it translocates through the pore. This control is achieved through biological motors, often *helicases*, which unwind the DNA and slow down the passage of one of the strands through the pore.

By applying a voltage current across the membrane, an ion current is maintained within the pore. In the absence of any molecules to be sensed, this current is referred to as the *base current.* When single-stranded DNA translocates through the pore, short DNA subfragments (approximately $3-5$ bases in length, referred to as k-mers, with $k = 3, 4, 5$) that fit into the pore cause a drop in the ion current as they block the movement of ions. The DNA is moves through the hole one base at a time, and the observation duration (dwell time) for the specific $3/4/5$-mer and the recorded current drop are used to estimate the sequenced DNA. The drop in current depends on the A, T, G, C content of the sequenced DNA. Generally, the current drop is influenced by the charge, 3D structure/shape of the nucleotides, and many other factors.

Similar to what is done with other sequencing technologies, each DNA fragment is replicated before sequencing to obtain multiple reads for reconstructing the original content. The reads corresponding to the same information string are generated via passage through different pores and/or at different times through the same pore. As a result, the reads may exhibit varying levels of sequencing noise. Typically, the process of deciphering the current readouts using multiple reads, known as "nanopore base calling," is facilitated by deep learning approaches involving convolutional and recurrent neural networks (CNNs and RNNs), described in more detail in [123].

Based on the previous discussion about the similarity of nucleotide chemical structures and the impact of k-mers on the ion current drop, it is evident that the accuracy of base calling in nanopores is expected to be lower compared to that of Illumina platforms. However, recent reports from ONT indicate significant progress in improving read reliability. According to ONT reports for R10.4 sequencing flowcells, the error rate for single molecule consensus is estimated at $> 0.1\%$. In academic labs, the observed error rates appear

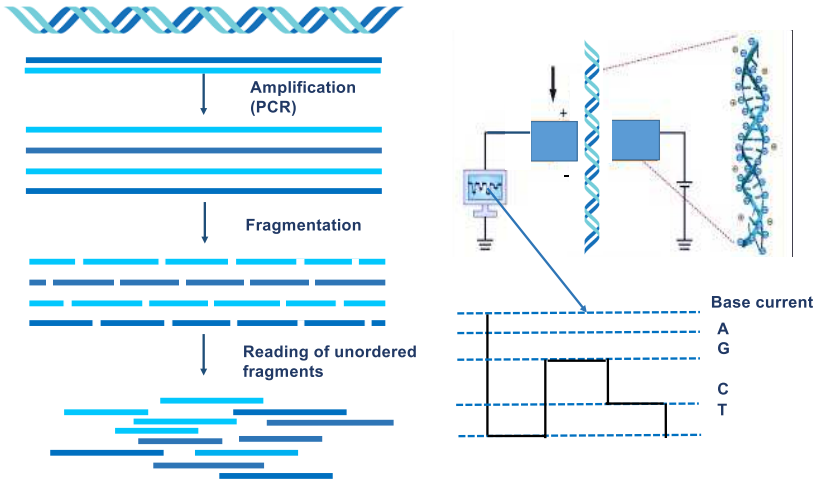


Fig. 4. Principles underlying next and third generation sequencing platforms: (left) shotgun sequencing, the idea behind sequencing long DNA strands broken into overlapping fragments that are stitched together during the assembly process (which can be thought off as finding fragments with long matching suffix-prefix strings). This approach is also the basic idea of the approach used in Illumina sequencing, along with the unique idea of *bridge amplification*; (right) a fundamentally different approach used in third generation ONT devices, termed nanopore sequencers. There, DNA strands are translocated (passed) through pores (holes), interrupting the flows of ions across the pore. The resulting drop in the ion current is indicative of the charge and structure of nucleotides within the pore.

to be significantly higher than 0.1%, with contributions from substitution, deletion, and insertion symbol errors.

The formation of consensus reads in nanopore sequencing is similar to the corresponding process in short-read technologies, but aligning ONT reads is computationally more challenging due to the presence of indel errors – see the description of *multiple sequence alignment algorithms* reported in the context of DNA-based data storage in [124], including Muscle, Coffee, Clustal Omega and others. The work [124] also introduced a specialized approach for error-correction from base-called reads using symbol-level redundancy, treating the problem as an instance of *trace reconstruction*. Trace reconstruction was initially described in the context of phylogenetic tree analysis [10] and is discussed in the context of coded trace reconstruction [24] in Section IV. We also remark that nanopore error-correcting codes that directly operate on raw current readouts without requiring intermediate basecalling are discussed in [17]. Some additional interesting results on reconstructing strings based on traces and modeling the nanopore channel can be found in [69], [71], and [75].

Although current DNA-based data storage systems do not broadly utilize PacBio HiFi technologies [60], it is important to highlight some notable features of this technology. HiFi sequencers produce long reads, ranging from $10,000$ to $20,000$ bases, and exhibit high reliability, comparable to that of Sanger sequencers. This increased accuracy in base calling can be attributed to various factors, including the reduction of polymerase bleaching effects and the implementation of subread consensus protocols. In the HiFi sequencing process, the same DNA molecule is read approximately 200 times, generating an equal number of subreads that are subsequently aligned and denoised. Unlike nanopores, HiFi devices capture the *kinetics* of the reading process, where the bases are characterized by distinguishable *random pulse widths,* and each pair of bases corresponds to different *random interpulse widths.* These pulse width and interpulse duration signals reflect the speed at which a polymerase incorporates a specific base into the subread. Our focus in subsequent discussions is exclusively on long-read nanopore-based DNA storage systems.

B. DNA Editing

DNA editing is an emerging interdisciplinary field with applications in chemistry, biology, medical sciences and synthetic biology, concerned with altering the content and structure of genomic and other -omic sequences. One of the major breakthrough discoveries in the area is the *CRISPR* (Clustered Regularly Interspaced Short Palindromic Repeats) system which was recently recognized by a Nobel prize in Chemistry awarded to its co-discoverers, Charpentier and Doudna [55]. CRISPR is a system native to some archaea and bacteria which use it as a form of immune and antiviral defense mechanism. The system involves repeat sequences of certain genetic sequences interleaved (interspersed) with spacer sequences that represent identifiers of invasive species encountered in the past. Upon detection of a recurrent invading unit through recognition of its characterizing genetic sequence, CRISPR's constituent Cas9 proteins guided by RNA recognition sequences cut the viral genomes at the position of the recognized content. Simply put, CRISPR stores snippets of genetic information of prior invasive species and uses this "genetic memory" to detect and disable present hostile viruses by cutting their genetic material (see Figure 5).

An advantage of CRISPR is that it is a complex that already involves enzymes such as Cas9 and relevant guide RNA sequences needed for disabling invasive species. Also, the complex performs cutting of single-stranded and double-stranded substrates in different manners. When cutting

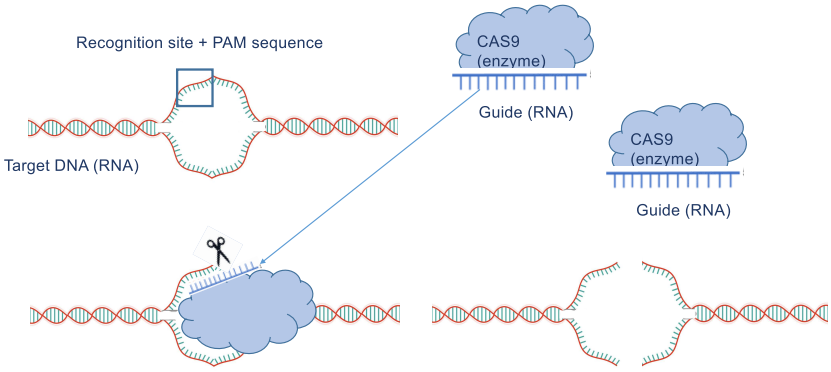


Fig. 5. The CRISPR system and its constituent Cas9 protein and guide RNA components. CRISPR memorizes genetic information of past invasive species and uses it to identify their renewed presence. Upon detection, it performs cutting of the genetic content of the infective agents in order to disable their replication. Outside of this native context, the complex can be used to cut arbitrary DNA strings through a careful design of the RNA guides.

double-stranded DNA, either both strands or only one of the two strands is cut. The latter process is usually referred to "nicking," and it does not lead to disassociation of the DNA duplex. There are also CRISPR complexes involving other enzymes, such as Cas13, which have the capability to edit RNA sequences.

For DNA-based data storage applications, and in particular, the subsequently discussed DNA Punchcards platforms, Cas9 can be matched with arbitrary synthesized guide RNA strings. These "lead" the enzymes to selected target positions, so that nicking may be performed in a massively parallel fashion involving multiple DNA sites. This is an especially important feature for molecular storage as it circumvents the problems associated with inherently sequential DNA synthesis: if nicks are to be introduced at multiple sufficiently distant locations in a double-stranded DNA substrate, the Cas9 enzymes can perform nicking without co-interference. One drawback of Cas9 is that it is what is known as a *single turnover* molecule – once the enzyme creates a nick it becomes inactive. This problem can be resolved by using *multiple turnover* enzymes (e.g., Pfago [115]) which, in principle, can make close to hundreds of cuts or nicks before becoming inactive.

C. Strand Displacement

Strand displacement in DNA is one of the most frequently used molecular and DNA computing paradigms. DNA strand displacement, as its name suggests, corresponds to replacing (part of) a single-stranded DNA section of a double-stranded DNA formation by another strand. There are two approaches to displacement: polymerase-based and *toehold-mediated*. We focus on toehold-mediated strand displacement as it is used more frequently, does not require specialized enzymes and lends itself to a wide variety of computations suitable for data stored in DNA Punchcards described in the next section [93].

The double-stranded DNA may be seen as encoding the state of a computational system or serve as a proxy for a logical gate. It comes with one or multiple single-stranded regions termed *toeholds,* which are usually of length $5-10$ bases. In Figure 6, there is one toehold in the right-most position of the double-stranded DNA that allows for hybridization of a

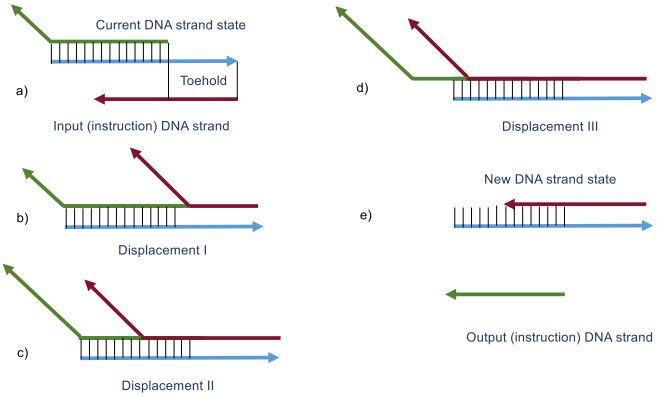


Fig. 6. Toehold-mediated strand displacement (with the displacement steps a), b) c), d) and e)). The input DNA strand hybridizes to the single-stranded toehold region and forces the competing strand to peel-off from the duplex, as governed by the laws of thermodynamics. Note that the green and red string are, by design, distinct, since they correspond to Watson-Crick complements of different parts of the string in blue. The reactions take milliseconds or less.

Watson-Crick-complementary single-stranded DNA, referred to as the *instruction strand*. Once the instruction strand hybridizes to the toehold it starts pushing out (i.e., displacing) the already present single-stranded part of the duplex to the left of the toehold until it completely disassociates. This strand then becomes the "output" of the computing unit. In a nutshell, the input strand may be seen as an instruction that changes the state and releases an output strand in its stead. Displacement reactions can be performed in a cascade, thereby allowing for multiple changes of states and released output strands which broadens the computational repertoire of strand displacement. As an example of the computations possible via strand displacement, the interested reader is referred to a neural network implementation based on cascades of toehold-mediated displacements [94]).

Although in theory many different computations, including universal ones, can be implemented via strand displacement, a major practical challenge is to control *leakage* in the cascades [116]. Leakage refers to unintended displacements that lead to the release of incorrect output strands and reduce the efficiency of the reactions. Leakage is the key impediment

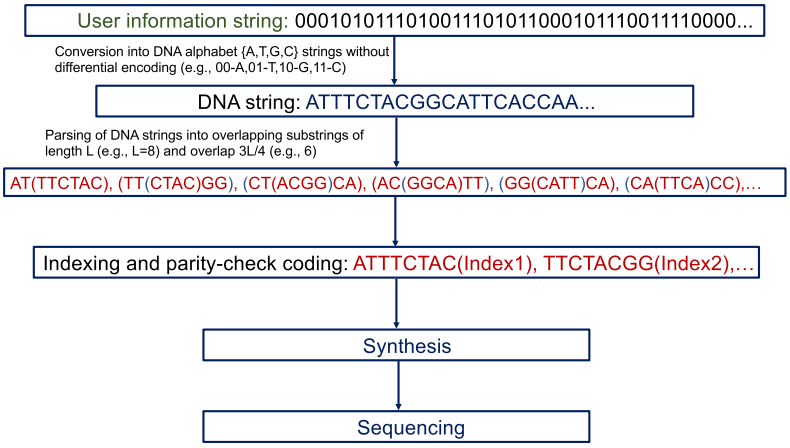


Fig. 7. The first two DNA-based data storage systems used nearly-identical data encoding protocols, involving runlength coding, indexing and single parity-check coding.

to accurate execution of more than $6-7$ consecutive displacement reactions, due to an excessive number of undesired byproducts. Recent methods for correcting leakage errors via controlled redundancy were described in [121].

III. An Overview of Existing DNA-Based Data Storage Platforms

The first successful implementations of DNA-based storage systems with read and write capabilities were described in [25] and [43]. These works outlined similar procedures, which involved the following steps.

a) Conversion of compressed binary data, such as text or images, into a ternary or quaternary alphabet. The process included elementary coding approaches such as GC-balancing, runlength coding, and single-parity check coding. Ternary encoding was employed to limit the runlengths of the same symbol (i.e., the lengths of homopolymers) to one. It effectively reduced the alphabet size from 4 to 3.

b) Parsing the encoded information strings into overlapping substrings with controlled overlap length. The overlap was set to 75%, ensuring 4-fold coverage of the content (with the exception of the two boundaries).

c) Addition of substring *identifiers* that encoded the index of each substring within the longer string. Note that such identifiers do not represent addresses since they were not designed to enable random access.

c) Synthesis of the overlapping substrings in the form of oligo pools.

d) Sequencing of the substrings and reconstructing the original message (refer to Figure 7).

The system was designed with several main considerations in mind. First, the overlapping oligo approach was used to facilitate easy reassembly of the original message through identification of overlapping suffixes and prefixes. As a side-effect, it effectively reduced the code rate to roughly $1/4$. Second, a balanced GC was ensured to reduce both synthesis and sequencing challenges. Third, since sequencing platforms in use at the time of the publication, such as Roche 454, were known to introduce errors in the presence of long homopolymer contents, the latter were severely constrained in length. However, it is important to note that current Next Generation sequencing platforms do not have severe homopolymer-related limitations, making it undesirable to compromise storage density through such a restrictive encoding protocol [113].

The DNA storage systems described above were unable to accurately reconstruct the original sequence, despite reducing the size of the alphabet and ensuring long overlaps between adjacent oligos. Furthermore, in order to access the encoded content in a specific section of the sequence, the user had to sequence and assemble the entire content, leading to significant implementation expenses.

The PCR-based random access approach was introduced and experimentally tested in [125], with its scalability further confirmed by Microsoft Research on a file size close to 200 MBs. The idea behind PCR-based random access is simple when analogies to classical storage systems are drawn: one needs to endow each information block (oligo) with an address sequence. The main challenge was to devise a method to efficiently search for the block with the required address when no "search circuits" are available. The obvious idea is to use hybridization, since the presence of a particular address sequence can be detected via targeted hybridization with its reverse complement sequence. This detection approach requires the desired strands to be *isolated* and *sequenced*. "Isolation" is achieved via amplification, i.e., PCR reactions. More precisely, the protocol for random access involves extracting a small subsample of the oligo mixture and running sufficiently many cycles of PCR reactions with primer combinations corresponding to the encoded information blocks to be retrieved. The amplified subsample in this case contains, with overwhelming probability, only the desired oligo content which can then be sequenced to complete the random access process.

The combinatorial design protocol for random access primers includes balancing the GC content, adding error-correcting redundancy, preventing self-folding of the primers, ensuring that pairs of distinct primers do not hybridize to each (i.e., preventing primer-dimers). Importantly, in addition

to all these constraints having to be met simultaneously, one more constraint has to be accounted for – zero *cross and autocorrelation* [46] of the primers. The autocorrelation of a binary string $s_1 s_2 \dots s_n$ is another binary string which indicates the overlap between prefixes and suffixes of the strings of lengths $1, 2, \dots, n-1$. Clearly, zero correlation prevents matching prefixes and suffixes of primers and thereby ensures some control over primer-dimer formations and self-folding. The concept is best illustrated by an example.

Example 1: Let $s_1 s_2 \dots s_5 = 10101$. *Then, the autocorrelation of the string is the binary string* 0101 *indicating that the prefix and suffix of the string of lengths* 3 *and* 1 *are the same (the second bit, equal to* 1*, indicates a match of length* 3*, while the fourth bit, again equal to* 1*, indicates a match of length* 1.

Cross-correlation can be defined similarly, by recording the prefix/suffix overlaps of two distinct strings. For more details, see [127] and the review article [126].

With regards to questions related to the enumeration of possible single-stranded DNA folds and related nonfolding constraints, the interested reader is referred to [78] and [87]. The latter work used the notion of *Motzkin paths* [88], which represent Dyck lattice paths augmented by flat (constant) platoes. A Dyck lattice path is a string of even length n over the alphabet $(,)$ containing exactly $n/2$ symbols of each type and satisfying the property that no prefix of the string contains more) than (symbols. A Motzkin path is a string of even length n over the alphabet $(,), -$ containing the same number of (and) symbols, and satisfying the property that no prefix of the string contains more) than (symbols (and with no restrictions on the placement of the symbols $-$). An example Dyck path of length 8 is $(()()())$, while an example Motzkin path of length 8 is $(-()-()-)$. The matched bracket symbols (and) can be used to represented paired bases within a string, while the dash $-$ symbol can be used to denote an unpaired basis. Restricted Motzkin paths described in [78] ensure that no short collection of consecutive unpaired bases (forming a loop) is followed by a long stem (a pair of reverse-complementary substrings on the string) which would lead to a stable secondary structure that renders the strings unusable as primers.

In addition to reporting the first PCR random access, the work reported in [125] also included a text rewriting scheme that is based on overlap-extension PCR; it also examined, from the theoretical point of view, how to perform information encoding so as to avoid substrings that are identical to the oligo primers used for addressing. In the context of rewriting, specialized encoding techniques were used to ensure that complete word phrases, likely to be edited together, are part of the same block that can be replaced by another block via overlap-extension PCR reactions. In the latter setting, a prefix-synchronized coding scheme adapted from [82] was used to ensure that the primer strings do not appear as substrings inside the information-bearing content, as that would lead to PCR amplification of substrands and not the whole oligo or gBlock. Sequencing was performed using Sanger methods, with no reported errors in the PCR-retrieved information.

Another important direction in DNA-based data storage was pursued in [124], where, for the first time, nanopore sequencing was used for data retrieval. The work also described how to combine the ideas of pilot signaling (from communication theory) and trace reconstruction [10] (from theoretical computer science) in DNA-based data storage. At the time the work was published (2015-2016), nanopore sequencers were the only low-cost and portable option for sequencing long blocks of DNA, such as gBlocks. Cost and portability are important issues given that Illumina platforms are bulky and expensive, and designed for lab use in mind. Despite their desirable properties, ONT MinION sequencers available in 2016 had the serious drawback of excessively high rates of indel errors, often exceeding 15% (this rate has been significantly reduced during the past decade, and is now closer to 5% for academic labs like the one used to perform the experiments in [124]).

A common approach to reduce the error rate is to form a consensus of all the nanopore readouts corresponding to different copies of the same input sequence. This naturally leads to the problem of *sequence alignment*, for which software suites such as Clustal Omega [106] or ONT in-house learning-based methods such as Nanopolish [123] are readily available. Still, for indel error-rates as high as 15%, the resulting consensus provided only a low-quality estimate for the actual user information string.

A straightforward solution to the problem was to treat the addresses as pilot sequences used to estimate the nanopore channel (see Figure 8). This approach proved successful since the addresses/pilots are indicative of "malfunctioning" or "tired" pores. For such pores, all traces or the most recently read traces contain a large number of errors. Given that the address sequences are known to both the encoder and decoder, the quality of the pore can be assessed through the number of errors in the addresses. By only using reads whose addresses have no errors, or by iteratively recruiting reads with low-error-rate-addresses to improve some local alignments, the reconstruction error rate dropped significantly, below $1-2\%$. The remaining errors were completely removed by GC-balancing the content of the blocks and by applying asymmetric homopolymer codes. The former allow one to identify potential synchronization or substitution errors by counting the symbols in each subblock; the latter allowed for fixing asymmetric deletion errors that affect one or two bases only, and do not completely erase a homopolymer. Such errors were found to occur in the ONT data generated by the experiments in [124]. Note that more recent ONT platforms, such as R10.4, are designed to accommodate $9-10$ bases in the pore in order to resolve the problem of homopolymer sequencing errors.

Two unconventional approaches to DNA-based data storage were explored in the recent studies [104], [115].

In the first approach, synthetic data was incorporated into the DNA of living organisms, such as bacteria. This *in vivo* (inside the cell) approach offers several advantages. First, user-defined information can naturally replicate itself through the growth of bacterial communities. Additionally, this population encoding strategy provides inherent error protection.

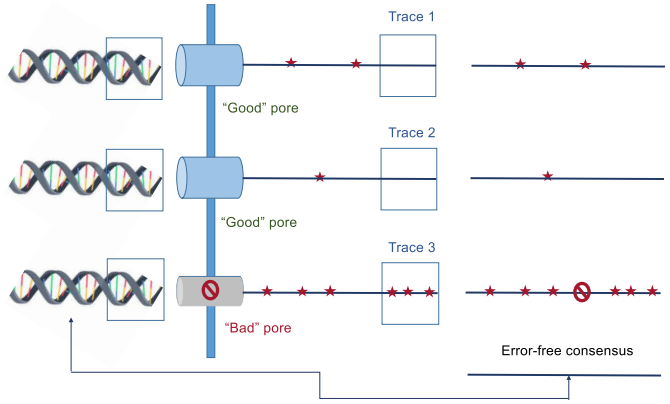


Fig. 8. The content of replicas of the same DNA string is read by different pores or by the same pore at different times. Each readout is modeled as a *trace* (which, unlike most traces used for mathematical proofs of reconstruction performance, also include insertions and substitution errors). The traces/reads can be aligned using one of the many existing DNA sequence alignment algorithms, but the results are of poor quality when even a small number of traces contains an excessive number of errors (see Extended Data Figure 3 in the Supplementary Information of [124]). Since the address sequences of the strings are known, the quality of the "nanopore channel" can be estimated by counting the number of errors present in the address string. In the given example, the address portion of the strings is boxed and the third trace contains three errors in the boxed region. This is indicative of a defective or tired pore and hence the third trace is not used for sequence alignment. As a result, the consensus sequence obtained via alignment of "good reads" is error-free or almost error-free.

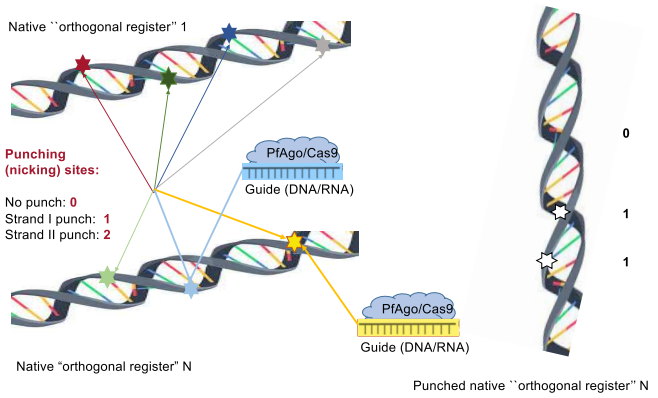


Fig. 9. In DNA Punchcard systems, data is recorded on native DNA fragments termed orthogonal registers. Each register has nicking locations whose sequence contexts are highly dissimilar, captured by the different colors of the stars indicating the nicking positions, thereby allowing for parallel nicking/recording of information on all orthogonal strands. The same nicking enzyme-guide unit can be used to nick hundreds of copies of the same register, which makes the system scalable.

However, this scheme has several drawbacks. The storage density is low compared to other methods due to the need to carefully place synthetic DNA in specific regions of bacterial genomes, so as not to disrupt normal cellular functions. Furthermore, the ratio between the information-bearing mass and the overall cell mass is significantly reduced as well, further decreasing the effective storage density. Most detrimentally, the process of recording and retrieving data is highly complex. Not only does one need to synthesize user DNA information, but must also insert it into desired locations within the bacterial genome. Data retrieval involves extracting bacterial DNA, isolating the desired content, and subsequently sequencing it. It remains uncertain whether this approach can be made cost-efficient enough to complete with purely synthetic *in vitro* (outside of the cell) methods.

In contrast, the DNA Punchcard system, introduced in [115], aims to address the issue of synthesizing DNA in the first place. The concept behind this approach is illustrated in Figure 9. In this storage context, "native DNA" refers to DNA extracted directly from bacteria, such as *E. coli*, without any synthetic modifications, and subsequently used and processed *in vitro*. Native DNA is readily available and can be obtained in large volumes at low cost. However, since native DNA has a composition determined by Nature, it cannot be easily altered to store user-defined data. Instead of modifying the content, one can instead choose altering the *topology* of the sugar-phosphate backbone at specific positions, termed "nicking positions." These positions are located between a pair of bases and indicate where the backbone strings are allowed to be nicked. Enzymes such as Cas9 or PfAgo, described in Section II, can be used for the recording process via nicking. The absence of a nick represents the value **0**, while a nick on the sense strand represents **1**, and the same nick on the antisense strand represents **2**. Therefore, the recoding alphabet in this system is ternary. It is important to note that cutting both strands at the same location is not allowed as it would cause the DNA to dissociate.

Storing information through nicking offers several advantages. First, there is no need to synthesize the information content in DNA, as topological changes are utilized to represent the data. Second, nicking can be performed in a highly parallel fashion. Third, writing the symbol **0** does not require any specific action, which is a characteristic shared by other existing storage technologies. Fourth, erasing and rewriting data is remarkably straightforward through the use of *ligases,* which can seal off the nicks. Since ligases remove nicks regardless of their position on the DNA, selective erasing and rewriting of data requires storing it in physically separated fragments of DNA.

The process of reading information stored in nicks is conceptually simple and highly robust to errors due to the existence of the bacterial genome reference. In a nutshell, the nicked DNA is denatured, i.e., the constituent strands are separated, and the obtained fragments PCR-amplified and sequenced using Illumina platforms. The sequenced reads are then aligned to the bacterial reference sequence to determine the locations were one fragment ended and another one started, corresponding to the positions that were nicked. To detect the nicks using nanopores requires reverting from ONT to solid-state nanopores, since the former do not require unwinding the strands [5].

Another important observation is that it is not necessary to use long native DNA fragments to encode information. This is because using long fragments can lead to undesired and off-targeted nicking. Instead, one can selectively isolate nonoverlapping fragments of native DNA that have low sequence similarity. These fragments are referred to as "orthogonal registers." By insisting on low sequence similarity,

measured in terms of the Levenshtein distance, the probability of nontargeted nicking is reduced while maintaining recording parallelism. Additionally, since the registers are substrings of a real bacterial genome, their order is determined by their occurrence in the genome. Therefore, there is no need for positional encoding.

Similarly to other current molecular recording systems, DNA Punchcards cannot avoid certain functional impairments. The most important drawback is reduced storage density, which is a consequence of both the decrease of the alphabet size from a 4-letter base alphabet to a 3-letter nicking alphabet, as well as the nicking site placement constraint. The latter requires nicking locations to be separated by roughly 10 bases apart in order to ensure stability of the DNA duplex and avoid having to read very short genomic fragments.

Note that the guides used in conjunction with the nicking enzymes still need to be synthesized unless they can be extracted directly from the native DNA itself without the use of other guides (which remains challenging). However, guides are typically very short RNA or DNA strings, of length $\leqslant$ 20 nucleotides. Furthermore, as already mentioned, the guides are multi-use entities when combined with enzymes like PfAgo.

Nick-based storage allows for in-memory computations to be performed directly and in parallel on the data recorded in all registers through strand displacement operations [22], [122]. In this computing approach, the symbols 0 and 1 are represented by two different blocks of bases, nicked at different locations. For example, if 5 nucleotides are used, 0 could be represented as $2-\mathtt{nick}-3$ while 1 could be represented as $4-\mathtt{nick}-1$, indicating that for the former, the nick is placed between the second and the third base, while in the latter, it is placed between the fourth and the fifth base. Since strand replacement terminates when the instruction string encounters a nick (as the nick prevents further "peeling-off" of a DNA substrate) and since nicks encode the bit values themselves, one can move the positions of the nicks around, thereby changing the register content. Roughly speaking, these nick-displacement operations involve sealing a nick in one position while creating a new nick in another position. Operations such as incrementing all registers, sorting their contents, and operations behind the universal Rule 110 automata have been successfully implemented and executed on data stored in DNA nicks via multistage strand displacement [22], [122].

Nick-based recording also enables the creation of 2D storage systems, as the nicks do not have to be necessarily superimposed on native DNA. The 2DDNA model of [89] superimposed nick-encoded data on synthetic DNA strands. Such an approach caters to the need of high-volume storage by encoding information in the DNA content and low-volume rewritable data storage by encoding it in the topological domain. Since the most prevalent data format is image data, the method was specialized to encode images into DNA content and image metadata (ownership information, date of access, steganographic messages) in the form of nicks. Another novel feature of this 2DDNA system is the use of machine learning methods to reconstruct the image in the presence of synthesis and missing oligo errors through a combination of automatic discoloration detection, image inpainting and smoothing (see Figure 10).

We conclude this review of different directions in DNA-based and molecular data storage by describing emerging approaches that aim to increase the size of the DNA alphabet through the use of chemical modifications [114] and employ synthetic polymers instead of synthetic DNA [61], [66].

In the former work, the DNA alphabet – A, T, G, C – is augmented by chemically modified native bases. A chemical modification is a small group of atoms added to a base so that it does not change its Watson-Crick binding affinity (or, at worst, does not significantly compromise it). The idea is to create "variants" of the symbol, say A_1, A_2, T_1, T_2, T_3, etc, that expand the alphabet size but remain distinguishable when sequenced. The main challenge of this approach is to adapt existing sequencing technologies – Illumina, ONT or PacBio – to efficiently discriminate all native and chemically modified symbols. The most promising approaches include learning to classify the bases using raw ion current signals from nanopores and kinetic information from PacBio SMRT (single-molecule, real-time) HiFi devices.[1] Another potential drawback of using chemically modified DNA is that PCR random access methods cannot preserve the information encoded in modification unless both strands contain "matching" chemical modifications. This problem can be remedied through the use of grids of self-rolled nanotubes that use the negative charge of the DNA sugar-phosphate backbone to control its movement via electronic circuits [57].

In the latter line of work, collections of synthetic polymers (usually two polymers, each assigned to one of the two bit values) of predetermined and largely different masses are connected to form bytes. Chemical bonds are introduced between the bytes to form one information-bearing string which, when broken, enables separate reading of each byte. Synthetic polymers offer the advantage of lower synthesis costs, although the synthesis process remains sequential. However, there are drawbacks, such as the absence of a PCR-type amplification process and a limited range of natural enzymes capable of working with the polymers. Initially, data retrieval from polymers relied on tandem mass spectrometry [66], but recent advancements have focused on the development and utilization of specialized nanopores [16].

IV. Coding-Theoretic Questions

As pointed out throughout the previous text, all components of different DNA storage systems introduce errors. For example, synthesis errors mostly manifest themselves in the form of substitution errors, while errors introduced during nanopore sequencing are standardly modeled as combinations of substitution and indel errors. In addition to these well-studied error models, many previously unexplored research

[1]The platforms operate by reading the *same molecule* $100-200$ times via synthesis and forming a consensus of the subreads to estimate the content of the molecule. Unlike nanopore sequencers that report ionic current signals, PacBio systems provide information about so-called pulse widths and inter-pulse durations, capturing the times taken by the polymerase to add a nucleotide and to prepare for adding the next one.

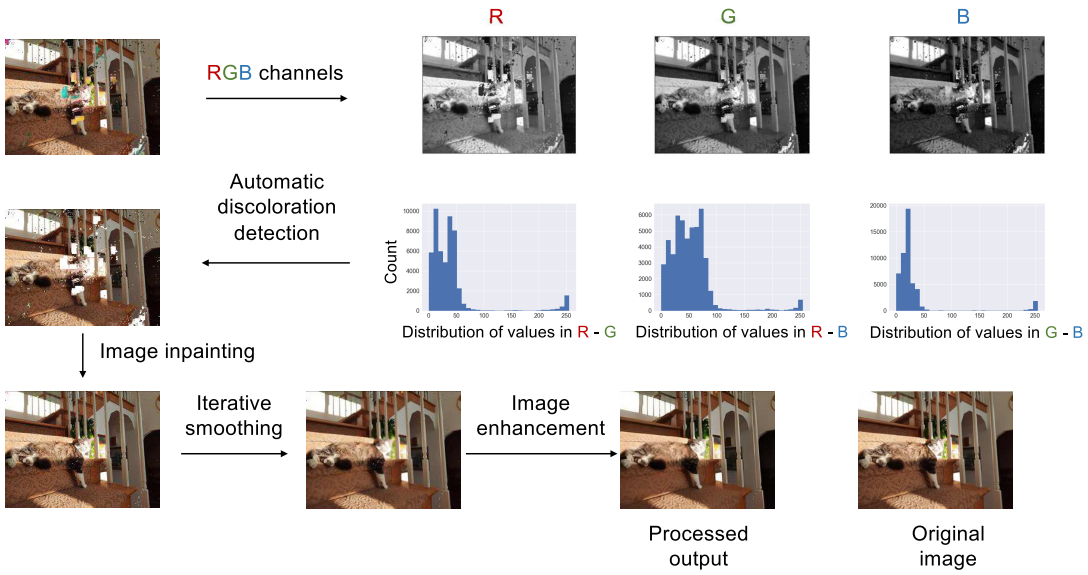


Fig. 10. An innovative component of the 2DDNA system is the use of automatic discoloration detection in images caused by synthesis and missing oligo errors. A specialized encoding scheme for image data separates the R,G, and B content of the images and places it on different oligos (top row). This allows for treating the three channels as a form of replication coding. Smoothness irregularities in one but not in the other two channels are indicative of oligo errors in the former (second row). These are detected by looking at the difference in values of pairs of color channels (as shown in the three histograms in the middle of the figure). The resulting discoloration is treated as missing pixels, whitened-out and then "imputed" using deep neural network inpainting methods. Further subjective image quality improvements are ensured via enhancement and smoothing (third row). Note that this approach is tailor-made for image data and it mitigates the use of costly error-correcting redundancy. For images with fine facial details, unequal error-protection low-density parity-codes can be used in addition to machine learning approaches to improve the reconstruction, with a redundancy of $< 7\%$ for the facial data alone.

directions in coding theory came into existence solely motivated by molecular storage. Some of these problems and their solutions were described in the review paper [126]. To avoid overlaps with the topics covered in [126], we choose to focus on a sampling of more recent analytic questions pertaining to modeling the DNA storage channel, decoding information via trace reconstruction and designing codes for DNA Punchcard systems.

The DNA storage model is an abstraction of a DNA-based data storage system that uses microarrays for DNA synthesis and Illumina and other short-read sequencing technologies, along with specialized graph-theoretic approaches for sequence reconstruction. Coded trace reconstruction is a new problem motivated by long-read nanopore sequencing approaches which require specialized alignment methods for data recovery. For DNA Punchcard storage systems which use known reference sequences and consequently have negligible readout errors, we choose to discuss coding problems related to duplex stability, rather than reconstruction errors. Finally, we also describe several problems in the area of *coding for unique reconstruction*, initiated by the works in [2], [38], and [58], which require constrained coding approaches that ensure unambiguous string recovery.

A. Coding for DNA Data Storage Systems With Short-Read Technologies

Our discussion to follow pertains to the first model of a DNA storage channel, described in [58]. It provides a simplified, yet conceptually accurate, abstraction of microarray-based synthesis and shotgun-type sequencing. To facilitate the mathematical exposition, we start with some relevant terminology.

Let n be a positive integer, $[q] = \{0, 1, \ldots, q-1\}$, and $\mathbf{x} \in [q]^n$. Choose a constant integer $0 < \ell$. The *ℓ-profile vector* of $\mathbf{x}$, denoted by $\pi_\ell(\mathbf{x})$, is a vector of length q^ℓ whose coordinates are indexed by all possible q-ary strings of length ℓ, in lexicographic order. The i-th entry of $\pi_\ell(\mathbf{x})$ equals the number of substrings of $\mathbf{x}$ that match the i-th string in the lexicographical order of strings in $[q]^\ell$. Note that the entries of the profile vector are nonnegative integers whose sum equals $n - \ell + 1$.

Example 2: If $q = 2$, $n = 5$, $\ell = 2$, *and* $\boldsymbol{x} = 11011$, *then* $\pi_2(\boldsymbol{x}) = 0112$, *and* $0 + 1 + 1 + 2 = 4 = 5 - 2 + 1$. *The lexicographical ordering used is* $(00, 01, 10, 11)$, *and the profile of* $\boldsymbol{x}$ *reveals that it contains no* 00 *substrings, that it includes exactly one substring* 01 *and* 10, *and two substrings* 11.

For simplicity of notation, we henceforth drop the subscript ℓ as it will be made clear from the context.

Next, we say that $\pi()$ is a *valid (string) profile* if there exists a string with that given profile. Otherwise, we say that the profile is not valid.

Example 3: For the parameters in the above example, $\pi = 2002$ *is not a valid profile, since there is no binary vector of length* $n = 5$ *that contains* 2 *substrings* 00 *and* 2 *substrings* 11.

Observe that two different string can share the same profile, as illustrated by the example that follows.

Example 4: Consider the following collection of strings

$$\{0000, 0010, 0100, 0110, 1001, 1011, 1101, 1111\}.$$

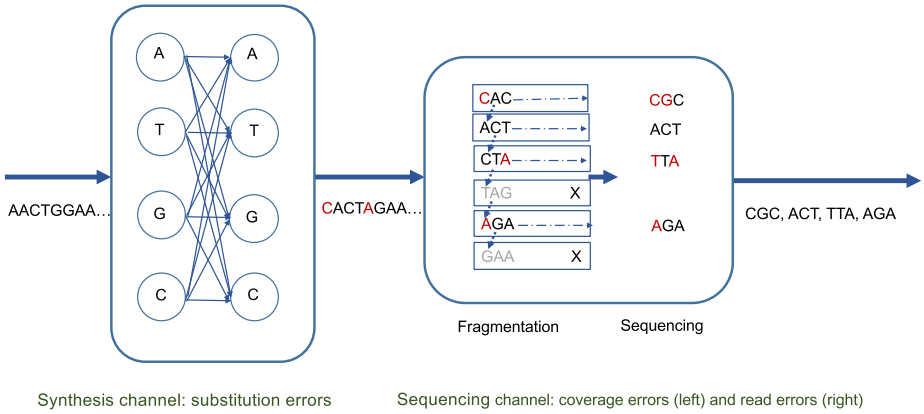


Fig. 11. A DNA-based data storage channel model. The input into the channel is a string of length n over the DNA alphabet $\{A, T, G, C\}$. The output of the channel comprises a collection of *noisy substrings* of the input string. Noise is introduced at three different stages of the write-read process. During synthesis (leftmost panel), one encounters *synthesis substitution errors*. In the particular example in the figure, such errors are marked in red, and they include the bases C and A. Note that synthesis errors propagate through the channel as they are "imprinted" into the string that is to be sequenced. Once the string is synthesized, it is read by first fragmenting it into ℓ-mers (in the example, 3-mers), some of which may be missing due to *coverage errors* (marked in gray). The substrings are then read through sequencing-by-synthesis, and the reading process itself can lead to the introduction of additional *sequencing substitution errors* within the substrings. Therefore, the input of the channel is a string, while the output of the channel is an incomplete, noisy collection of substrings of the input string.

The $\ell = 2$-mer equivalence classes of the strings, with two strings being equivalent if their $\ell = 2$-mer profile vectors are the same, are

$$\{0000\}, \{0010, 0100, 1001\}, \{0110, 1011, 1101\}, \{1111\}.$$

Clearly, the profile vectors of the four equivalence classes are $(3, 0, 0, 0)$, $(1, 1, 1, 0)$, $(0, 1, 1, 1)$ *and* $(0, 0, 0, 3)$, *respectively.*

Next, for profile vectors of two q-ary strings $\mathbf{x}$ and $\mathbf{y}$, let us define their *asymmetric profile distance* according to

$$\Delta(\mathbf{x}, \mathbf{y}) = \max\{\partial(\mathbf{x}, \mathbf{y}), \partial(\mathbf{y}, \mathbf{x})\},$$

where $\partial(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{q^\ell - 1} \max\{\pi(\mathbf{x})_i - \pi(\mathbf{y})_i, 0\}$, and where the subscript i denotes the i-th coordinate of the vector.

The question of interest is to design the largest possible codebook $\mathcal{C}_{q,n,\ell,d}$ of q-ary vectors of length n such that the minimum pairwise asymmetric distance of their ℓ-profile vectors is at least d. For $d \geqslant 1$, one has to automatically preclude simultaneous inclusion of two strings from the same equivalence class in $\mathcal{C}_{q,n,\ell,d}$, since in that case, the profiles of the strings are the same.

The motivation for studying the previously introduced problem comes directly from the models of DNA-based data storage depicted in Figure 11. When sequencing DNA, the input to the sequencer is a string, while the output is a collection of substrings (reads) generated (for this model) by Illumina sequencers. The asymmetric distance allows one to account for three types of errors, *synthesis substitution errors,* whose number is assumed to be upper-bounded by s; *coverage errors,* modeled as missing substrings, the number of which does not exceed c; and, *sequencing substitution errors,* which manifest themselves as individual substitution errors within the substrings (reads), and the number of which does not exceed o. If the minimum asymmetric distance of the profiles of the codestrings satisfies $d_{min} \geqslant 2s + c + o$, then the code can correct the corresponding number of synthesis, coverage and sequencing errors. Several abstractions are made to make the analysis of this model tractable. First, it is assumed that one can perfectly count distinct oligos. In the example with $\mathbf{x} = 11011$ and $\pi(\mathbf{x}) = 0112$, it is assumed that one can determine that there were *two distinct* substrings 11 in the original string. In practice, Illumina systems do actually report all sequenced oligos, but without the information if these oligos are replicas of the same substring or replicas of multiple identical substrings. This issue can be mitigated through the use of long-read technologies which are known to resolve problems associated with repeats. Second, synthesis errors are usually context dependent, and repeats make the process difficult or outright impossible. To make the model more realistic, we would require the codestrings to be repeat-free, but this would make the subsequent analysis very hard. Third, since multiple replicas of the same string may be generated during sample preparation, and each of these strings can be subject to different error-patterns, one substring can give rise to multiple erroneous substrings. How many replicas are present depends on the coverage depth (i.e., the average number of times a symbol is covered by the reads).

Given that the DNA storage channel accepts *strings* at the input and produces profiles at the output, it is not immediately clear how to ensure that minimum asymmetric distance constraints are met in the substring domain while working with the global input strings. The key ideas for solving this problem rely on the use of **de Bruijn graphs** [12], [15] and are described next.

A *directed graph (digraph)* D is a pair of sets (V, E), where V is the set of *nodes (also referred to as vertices)* and E is a set of ordered pairs of V, termed *arcs*. If $e = (v, v')$ is an arc, we call v the *initial* node (tail) and v' the *terminal* (head) node. We allow loops (i.e., we allow $v = v'$) as well as multiple arcs between nodes.

The *incidence matrix* of a digraph $\mathbf{D}$ is a matrix $\mathbf{B}(\mathbf{D}) \in \{-1,0,1\}^{\mathsf{V}\times\mathsf{E}}$, where

$$\mathbf{B}(\mathbf{D})_{v,e}=\begin{cases}1, & \text{if } e \text{ is not a loop and } v \text{ is its terminal node,}\\ -1, & \text{if } e \text{ is not a loop and } v \text{ is its initial node,}\\ 0, & \text{otherwise.}\end{cases}$$

Given q and ℓ, the (standard) de Bruijn graph is defined on the node set $[q]^{\ell-1}$, where we recall that $[q]=\{1,\ldots,q\}$. For $\mathbf{v},\mathbf{v}'\in[\![q]\!]^{\ell-1}$, an ordered pair $(\mathbf{v},\mathbf{v}')\in\mathsf{E}$ if and only if $v_i=v'_{i-1}$, for $2\leqslant i\leqslant \ell-1$. We label the arc $(\mathbf{v},\mathbf{v}')$ with the length-ℓ string $\mathbf{v}v'_{\ell-1}$, and without loss of generality, equate arcs with their labels.

Example 5: Let $q=2$ *and* $\ell=4$ *and consider the de Bruijn graph shown Figure 12. The nodes* $\mathbf{v}=101$ *and* $\mathbf{v}'=010$ *are connected by the arc* 1010 *which originates from* $\mathbf{v}$ *and terminates in* $\mathbf{v}'$. *The suffix of* $\mathbf{v}$ *of length* $\ell-2=2$ *equals* 01, *which is also the prefix of length* $\ell-2$ *of* $\mathbf{v}'$. *The label of the arc equals* $\mathbf{v}v'_{\ell-1}=1010$.

The notion of de Bruijn graphs can be extended to prohibit the presence of certain ℓ-mer arc labels or $(\ell-1)$-mer vertex labels [98]. For such *restricted de Bruijn graphs,* the set of allowed $(\ell-1)$-mers is denoted by S. The corresponding restricted de Bruijn graph is denoted by $D(S)$. The importance of restricted de Bruijn graphs for DNA-based storage systems lies in the fact that S may be chosen to satisfy additional sequence constraints, such as balanced GC constraint (e.g., balanced $\ell-1$-mers). For $q=2$, "balanced" refers to the substrings containing the same number of 0s and 1s, while for larger values of q, it refers to balanced or nearly balanced GC content.

A *walk* of length n in a digraph is an ordered collection of nodes, $v_0v_1\cdots v_n$, with $(v_i,v_{i+1})\in\mathsf{E}$ for all $i\in[\![n]\!]$. A walk is *closed* provided that $v_0=v_n$. A *cycle* is a closed walk with no repeated nodes, i.e., $v_i\neq v_j$, for $0\leqslant i<j<n$. A cycle of length one is referred to as a loop. Given a subset A of the arc set, let $\mathbf{a}\in\{\mathbf{0},\mathbf{1}\}^{|\mathbf{E}|}$ be its *incidence vector*, so that $\mathbf{a}_e=1$ if $e\in\mathsf{A}$ and $\mathbf{a}_e=0$ otherwise. For the incidence vector a of a closed walk in D, we have $\mathbf{B}(\mathbf{D})\mathbf{a}=\mathbf{0}$.

The de Bruijn digraphs of interest to our problem have arcs weighted by nonnegative integers that reflect the properties of a chosen sequence that they represent. More precisely, the weight of an arc indicates the count of the substring label within the sequence. As an example, in Figure 12, only three arcs have integer labels marked in black. All arcs without integer labels are assumed to have weight zero. Since the label of each arc is uniquely determined by the source and terminal vertex, one can omit the sequence label and only retain the arc weights. The weights in the figure describe how many times an arc has to be traversed and simultaneously, they capture the number of times the substring appeared in the string (i.e., they capture the profile of a string).

Example 6: One of the possible strings whose profile is shown in the de Bruijn graph example equals 1001001, *since it contains the following substrings of length* $\ell=4$*:* $\{1001,0010,0100,1001\}$. *Hence, to recover the string, the arc labeled* 1001 *has to be traversed twice, while the arcs labeled* $\{0010,0100\}$ *have to be traversed once. All other arc have to be ignored. The length of the string is* $n=4+\ell+1=7$, *since there are* 4 *substrings in total. The length of the walk in the graph equals the sum of the arc labels or the number of substrings of the string, which equals* $n-\ell+1=4$.

A *constrained* walk from some node $\mathbf{v}$ to another node $\mathbf{v}'$ in $\mathbf{D}(S)$ describes a string that starts with $\mathbf{v}$ and ends with $\mathbf{v}'$ and whose ℓ-mers are restricted to belong to S. *Closed strings* are strings that start and end with the same $(\ell-1)$-mer and they correspond with closed walks in $\mathbf{D}(S)$. Strings corresponding to walks of length $n-\ell+1$ in $\mathbf{D}(S)$ (and, consequently, profiles of strings of length n) are denoted by $\mathcal{Q}(n;S)$. At the same time, the set of *closed strings* is denoted by $\bar{\mathcal{Q}}(n;S)$, and clearly, one has $\bar{\mathcal{Q}}(n;S)\subseteq\mathcal{Q}(n;S)$. The set of profile vectors of closed strings is denoted by $\mathbf{p}\bar{\mathcal{Q}}(n;S)$. The reason for introducing closed strings is that for such strings, several counting problems simplify substantially, while the restriction has barely any bearing on the code rate for constant ℓ.

Suppose that $\mathbf{u}\in\mathbf{p}\bar{\mathcal{Q}}(n;S)$. Then, the *flow conservation equations* below hold.

$$\mathbf{B}(\mathbf{D}(S))\mathbf{u}=\mathbf{0}. \tag{1}$$

Furthermore, let $\mathbf{1}$ denote the all-ones vector. Since the number of ℓ-mers in a string of length n equals $n-\ell+1$, we also have

$$\mathbf{1}^{\mathsf{T}}\mathbf{u}=n-\ell+1, \tag{2}$$

where T denotes the transpose. Let $\mathbf{A}(S)$ be $\mathbf{B}(\mathbf{D}(S))$ augmented with a top row $\mathbf{1}^{\mathsf{T}}$; also, let $\mathbf{b}$ be a vector of length $|V(S)|+1$ with a one as its first entry, and zeros elsewhere. Equations (1) and (2) may then be jointly rewritten as

$$\mathbf{A}(S)\mathbf{u}=(n-\ell+1)\mathbf{b}.$$

Consider next the following two sets of integer points.

$$\mathcal{F}(n;S)\triangleq\{\mathbf{u}\in\mathbb{Z}^{|S|}:\mathbf{A}(S)\mathbf{u}=(n-\ell+1)\mathbf{b},\ \mathbf{u}\geqslant\mathbf{0}\}, \tag{3}$$

$$\mathcal{E}(n;S)\triangleq\{\mathbf{u}\in\mathbb{Z}^{|S|}:\mathbf{A}(S)\mathbf{u}=(n-\ell+1)\mathbf{b},\ \mathbf{u}>\mathbf{0}\}. \tag{4}$$

It is straightforward to see that the profile vector of any closed string must belong to $\mathcal{F}(n;S)$. Conversely, any vector in $\mathcal{E}(n;S)$ is a profile vector of some closed string.

The above formulation can be used to establish a count of the number of profile equivalence classes as follows. Suppose that $D(S)$ is strongly connected. Then, under certain mild constraints and for a constant values of ℓ, it can be shown that $|\mathcal{E}(n;S)|\sim n^{|S|-|V(S)|}$ and $|\mathcal{F}(n;S)|\sim n^{|S|-|V(S)|}$; as a result, $|\mathbf{p}\bar{\mathcal{Q}}(n;S)|\sim n^{|S|-|V(S)|}$. Here, the symbol $\sim$ is used to indicate that for sufficiently large n, the sizes of the sets scale as the term on the right.[2] In a nutshell, the result follows by counting the solutions of the defining conditions for points in $\mathcal{F}(n;S)$ and $\mathcal{E}(n;S)$ via *lattice point enumeration techniques* and what is known as Erhart-McDonald's reciprocity theory. The Erhart-McDonald's reciprocity theory is a broad generalization of a simpler result known as *Pick's theorem,* which expresses the

[2]A rigorous statement of the results involves the definition of quasipolynomials and is therefore omitted.

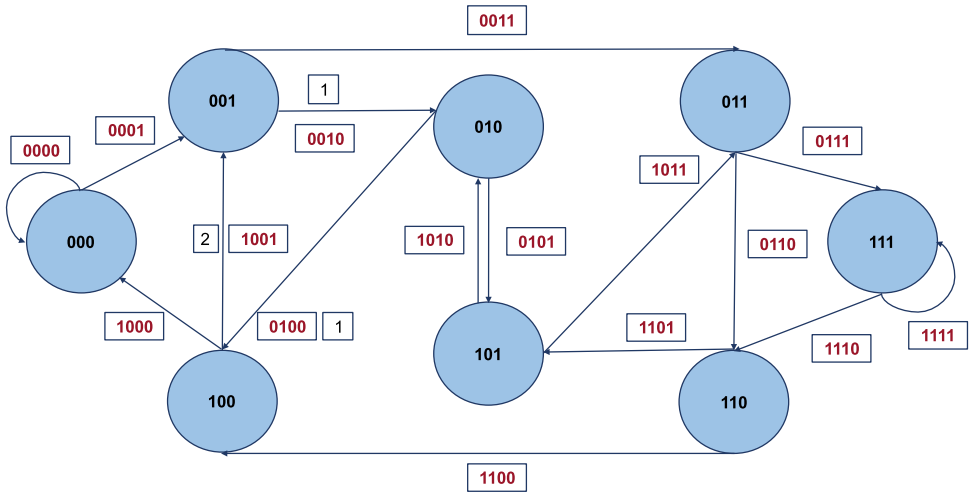


Fig. 12. The de Bruijn graph for $q = 2$, $\ell = 4$. Vertex labels are binary vectors of length $\ell - 1$, while arc labels are binary vectors of length ℓ. The nonnegative integer weight of arcs describe the $\ell = 4$-mer profile vector of the string $\mathbf{x} = 1001001$, $\pi(\mathbf{x}) = 00101000002000000$. Unweighted arcs are assumed to have weight 0.

area of a polygon in terms of the number of lattice points in its interior [92].

For unrestricted de Bruijn graphs, which are strongly connected, we have $n^{|S|-|V(S)|} = n^{q^\ell - q^{\ell-1}}$. This expression is the asymptotic for the number of *distinct* ℓ-gram profiles of q-ary strings of length n. The result also establishes that $|\mathcal{C}_{q,n,\ell,d=1}| \sim n^{q^\ell - q^{\ell-1}}$.

To determine $|\mathcal{C}_{q,n,\ell,d}|$ for $d > 1$, we need to ensure that the profiles are not only distinct but at a asymmetric distance $\geqslant d$ from each other. This can be ensured by adding more constraints to the profile vectors (i.e., in addition to the flow- and sum-constraints) that also take the form of linear equations. The solution involves using *Varshamov codes* [120], designed specifically for asymmetric channels. For convenience, we describe these codes below.

Fix a positive integer d, and let p be a prime such that $p > \max\{d, N\}$ (where, for notational convenience, we used N to denote $|S|$). Next, choose N distinct nonzero elements $\alpha_1, \alpha_2, \ldots, \alpha_N$ in $\mathbb{Z}/p\mathbb{Z}$ and let

$$\mathbf{H} \triangleq \begin{pmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_N \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_N^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^d & \alpha_2^d & \cdots & \alpha_N^d \end{pmatrix}.$$

Pick any vector $\boldsymbol{\beta} \in (\mathbb{Z}/p\mathbb{Z})^d$ and define a code according to

$$\mathcal{C}(\mathbf{H}, \boldsymbol{\beta}) \triangleq \{\mathbf{u} : \mathbf{H}\mathbf{u} \equiv \boldsymbol{\beta} \bmod p\}. \tag{5}$$

Then, $\mathcal{C}(\mathbf{H}, \boldsymbol{\beta})$ is an asymmetric error-correcting codes of length N with (designed) minimum asymmetric distance $d+1$. Hence, all the codestrings of a Varshamov code that are valid profile vectors are also d-asymmetric-error-correcting codestrings. More precisely, we can construct profile codes with $|\mathcal{C}_{q,n,\ell,d}| \sim |\mathcal{C}(\mathbf{H}, \boldsymbol{\beta}) \cap \mathbf{p}\mathcal{Q}(n; S)|$, for all $\boldsymbol{\beta} \in (\mathbb{Z}/p\mathbb{Z})^d$. By invoking the pigeon-hole principle, we can show that there exists a $\boldsymbol{\beta}$ such that $|\mathcal{C}(\mathbf{H}, \boldsymbol{\beta}) \cap \mathbf{p}\mathcal{Q}(n; S)| \geqslant |\mathbf{p}\mathcal{Q}(n; S)|/p^d$. The choice of $\boldsymbol{\beta}$ that maximizes the code size is not known in general, but this is not a significant practical issue. Furthermore, $|\mathcal{C} \cap \mathbf{p}\mathcal{Q}(n; S)|$ is typically strictly smaller than $|\mathcal{C}|$, and deriving analytical bounds for the code size is nontrivial (see [58] for details).

Suppose next that $\mathcal{C}$ is a Varshamov asymmetric distance error-correcting code with parameters N, d. We construct DNA profile codes from $\mathcal{C}$ as follows.

1) When $N = |S|$, we use the intersection of $\mathcal{C}$ and $\mathbf{p}\mathcal{Q}(n; S)$ as our ℓ-gram asymmetric error-correcting code. Simply put, we choose the codestrings in the Varshamov code $\mathcal{C}$ that are also profile vectors.
2) When $N < |S|$, we extend codestrings in $\mathcal{C}$ to profile vector of length $|S|$ in $\mathbf{p}\mathcal{Q}(n; S)$. Note that one may not always be able to extend an arbitrary string to a profile vector.

Example 7: Let $q = 2$, $\ell = 3$, $S = \{001, 010, 011, 100, 101, 110\}$ *so that* $N = 6$. *Note that the strings in* S *are as closed to balanced as possible, since* S *does not include* 000 *nor* 111. *Let* $d = 3$ *and choose* $p = 7$, *so that*

$$\mathbf{H} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 4 & 2 & 2 & 4 & 1 \end{pmatrix}, \text{ and let } \boldsymbol{\beta} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Then, $\mathcal{C}(\mathbf{H}, \boldsymbol{\beta})$ *contains the following strings.*

$$\begin{array}{ll} (\mathbf{0,0,2,0,2,2}) \leftrightarrow \mathbf{01101101} & (\mathbf{0,1,1,4,0,0}) \\ (\mathbf{2,2,0,2,0,0}) \leftrightarrow \mathbf{00100100} & (\mathbf{0,1,0,0,4,1}) \\ (1,4,0,0,1,0) & (0,0,4,1,1,0) \\ (\mathbf{1,1,1,1,1,1}) \leftrightarrow \mathbf{00101100} & (\mathbf{4,0,0,1,0,1}) \\ (1,0,1,0,0,4) & \end{array}$$

Of these Varshamov codestrings, only three (marked in boldface letters) are valid profile vectors from $\mathbf{p}\mathcal{Q}(8; S)$. *Hence, for the chosen set of parameter values, our codebook would include three allowed profile codestrings, and consequently, three input codestrings (where we are allowed to select one string representative from each allowed profile class).*

Two final observations are in place.

First, counting the codestrings in a Varshamov-type profile code once again, like in the equivalence class counting framework, reduces to computing the lattice point enumerator of the intersection of the lattices defined by $\mathbf{A}(S)$ and $\mathcal{C}(\mathbf{H}, \boldsymbol{\beta})$. Finding lattice point enumerators is a fundamental problem in discrete optimization and high-quality software suites for solving the problem are available. One such software, `LattE`, reported in [7], is based on an elegant algorithm described in [9] that triangulates the supporting cones of the vertices of a polytope to obtain simplicial cones which are then recursively decomposed into unimodular cones. The algorithm performs enumeration of lattice points in polynomial time whenever the dimension of the polytope is fixed.

Second, as already pointed out, the asymmetric code construction procedure is implemented to produce profile vectors, which are actual outputs of the channel and not the desired input codestrings. We hence need to convert the profiles back into strings, with exactly one string corresponding to one profile. This can be accomplished by once again using de Bruijn graphs that capture both the substrings and their multiplicities: all that is needed is to find a path in the graph that traverses each arc a number of times indicated by its weight multiplicity. This is akin to the process of sequence assembly that is widely used in computational biology [84]. For related results which accommodate a larger range of parameter values, please refer to [18].

B. Coded Trace Reconstruction

The problem of trace reconstruction was introduced in [10], motivated by sequence analysis problems first considered by Levenshtein [64], as well as practical sequence alignment questions in phylogeny and computational biology. The relevance of the problem to DNA-based data storage comes through its connection to sequence alignment, which is necessary when reading the information content via nanopore sequencers. In Section III, we discussed various sequence alignment algorithms that originated from computational biology that can be used to create consensus sequences from multiple noisy reads. Most of these methods rely on dynamic programming approaches, and are therefore hard to analyze. Trace reconstruction, on the other hand, is an abstraction that is conceptually simple to state and understand, and which comes with a cohort of straightforward reconstruction algorithms. Nevertheless, trace reconstruction is also coupled with nontrivial, but tractable, analytical challenges (see also [110]).

In a nutshell, the trace reconstruction problem asks how many noisy copies (reads) are needed to reconstruct a string with high probability. More formally, the assumption is that there exists an unknown string $\mathbf{x} \in \{0,1\}^n$, and that one is given access to *traces* of $\mathbf{x}$, which are generated by passing $\mathbf{x}$ through a deletion channel (e.g., a nanopore modeled as a deletion channel [124]). The deletion channel independently deletes bits of $\mathbf{x}$ with a given deletion probability δ, and each pass through the channel produces a trace which is independent from all other traces. Clearly, traces represent subsequences of $\mathbf{x}$ of varied length formed in a probabilistic, i.i.d manner. The formal goal is to minimize the number of traces, i.e., the number of reads, that need to be acquired in order to reconstruct $\mathbf{x}$ with high probability.

Note that a solution for the binary trace reconstruction problem automatically leads to solutions for nonbinary settings. The precise statement is provided below, as stated in [76].

If T traces suffice to reconstruct a random string in $\{0,1\}^n$ with probability at least $1-\gamma$, then T traces also suffice to reconstruct a random string in $\{1,\ldots,q\}^n$ with probability at least $1 - O(\gamma \log q)$.

This is the reason why we mostly focus on results for binary strings. Towards the end of the section, we explain in more detail how coded binary strings can be translated into coded quaternary strings.

The focus of the trace reconstruction research area has been mostly on two types of approaches: *worst-case* [29], [86], where the requirement is for the reconstruction procedure to work for all strings in $\{0,1\}^n$, and *average-case* [48], [50], [91], where the reconstruction algorithm is only required to work with high probability for a string selected uniformly at random. Formally, worst-case trace reconstruction is concerned with designing a reconstruction algorithm, $\mathcal{R}$, such that *for every* $\mathbf{x} \in \{0,1\}^n$ one has

$$P_{T_1,\ldots,T_t}\left[\mathcal{R}(T_1,\ldots,T_t) = \mathbf{x}\right] \geqslant 1 - 1/n,$$

where T_i, $i \in [t]$, stands for traces of $\mathbf{x}$ which are i.i.d. with respect to the output distribution of a deletion channel with deletion probability δ. The goal is to make $t = t(n)$ as small as possible. Note that we use the lower bound $1-1/n$ on the reconstruction probability following [24], as this bound allows one to avoid notational clutter and simplify the expressions for our main results. In the second part of the section, we extend this definition by replacing n with an arbitrary polynomial in n. For average-case reconstruction, we require that $\mathcal{R}$ satisfy

$$2^{-n} \sum_{\mathbf{x}\in\{0,1\}^n} P_{T_1,\ldots,T_t}\left[\mathcal{R}(T_1,\ldots,T_t) = \mathbf{x}\right] \geqslant 1 - 1/n, \qquad (6)$$

where we would once again like to make $t = t(n)$ as small as possible, and where the traces T_i, $i \in [t]$, have the same properties as stated for the worst-case problem. It is clear that the number of traces required for average-case trace reconstruction is smaller than that required for worst-case trace reconstruction.

The state-of-the-art results for average-case reconstruction [50] established that $\exp(O(\log^{1/3} n))$ traces suffice to reconstruct a random n-bit string under arbitrary *constant* deletion probability δ. Handling the worst-case setting is significantly more challenging. The currently best upper bound [19] equals $\exp(O(n^{1/5}))$, improving the $\exp(O(n^{1/3}))$ result of [29] and [86] based on an algorithm that exploits

single-bit statistics of the traces.[3] The gap between the upper and lower bound is still prohibitively large. The state-of-the-art lower bounds [20] are roughly $\log^{5/2} n \log\log^{-7} n$ traces for average-case trace reconstruction and $n^{3/2}\log^{-7} n$ traces for the worst-case setting. For some more recent results regarding k-mer statistics approaches, please refer to [23] and [74].

Unlike other applications, DNA-based data storage allows choosing a subset of strings with desirable properties to be used for trace reconstruction-based alignment. For example, one can focus on fixed weight strings, strings that satisfy bounded runlength (homopolymer) properties, balanced GC contents and others. This more restricted sequence selection process naturally leads to the question of *coded trace reconstruction* [24]. Here, the goal is to design codes with asymptotic rate equal to or close to 1 that are also efficiently encodable and decodable using significantly fewer traces than needed for the unrestricted (uncoded) setting. As in all other works, the assumption is that we work with constant channel deletion probabilities. A simple yet significantly more parameter-restricted line of work addressed the coded trace reconstruction problem for a *constant number* of deletions, using concatenations of Varshamov-Tenengolts codes [1]. Another line of work [14] built upon the techniques of [24] and provided improvements on the number of traces required as a function of the rate. A small drawback of the latter method is the need for preprocessing, which requires superpolynomial time.

To better understand our approach to coded trace reconstruction, let us revisit the ideas from [124] which for the first time modeled the nanopore sequencing process as trace reconstruction. The codestrings were designed to satisfy block-wise GC-balancing constraints, with each block of 8 symbols over the alphabet $\{A, T, G, C\}$ perfectly balanced. Balancing constraints were used to ensure correct synthesis, but somewhat serendipitously proved useful for trace reconstruction. The utility of balancing for string reconstruction is in part due to the related runlength constraints. Block-based balancing also allows symbol runlength (homopolymer) constraints to be automatically satisfied. The traces obtained via nanopore sequencing in [124] were used to form a consensus sequence, which was then updated in several iterations by checking if the block-level balancing constraints are met. For simplicity, we will illustrate the underlying ideas through a simple adaptation of the Bitwise Majority Alignment (BMA) algorithm [10], although this algorithm does not perform as well as the actual algorithm used in [124] due to not being able to handle context-dependent indels and substitution errors.

Example 8: Let the codestring to be sequenced by nanopores equal

$$\boldsymbol{s} = \mathtt{AATGGCGA\ TTCCGGAA\ GGGAATCA},$$

comprising three blocks of length 8*, each with a perfectly balanced (*50%*)* GC *content (note that the string is parsed into blocks for ease of visualization). Now, assume that the sequencer produced* 5 *reads/traces based for the input string* $\boldsymbol{s}$*, as listed below.*

ATGGCGTTCGGAAGGATCA
AATGGTTCCGGAAGGAAT
AATGGCGATTCCGGGGGAAA
GCGATTCCGGGGAATA
ATGGATCGAGGATCA.

The algorithm proceeds by constructing the consensus sequence by focusing on one position at the time, finding the majority symbol and calling it the consensus symbol, and then shifting the mismatched symbols one position to the right. The first three steps of the approach applied to the above DNA strings are presented below, along with the final consensus result. Ties are broken arbitrary but recorded for subsequent re-examination. Majority symbols are written in boldfaced letters, while minority symbols are replaced by "-" and moved to the right. Note that in Step 2, the tie is broken in favor of A*, but both symbols* A *and* T *are recorded for subsequent consideration.*

Step 1:

ATGGCGTTCGGAAGGATCA
AATGGTTCCGGAAGGAAT
AATGGCGATTCCGGGGGAAA
-GCGATTCCGGGGAATA
ATGGATCGAGGATCA
A.

Step 2:

A**-**TGGCGTTCGGAAGGATCA
AATGGTTCCGGAAGGAAT
AATGGCGATTCCGGGGGAAA
–GCGATTCCGGGGAATA
A**-**TGGATCGAGGATCA
AA
A(**A**/**T**).

Step 3:

A**-**TGGCGTTCGGAAGGATCA
AATGGTTCCGGAAGGAAT
AATGGCGATTCCGGGGGAAA
–-GCGATTCCGGGGAATA
A**-**TGGATCGAGGATCA
AAT
A(**A**/**T**)**T**.

The original sequence, the consensus sequence, and the consensus with ties are summarized below, respectively. Mismatches are indicated in red, and the sequences parsed into groups of 8 *symbols for future analysis.*

AATGGCGA TTCCGGAA GGGAATCA
AATGGCGA TTCCGGAG GAGGATACAT

[3]The latter upper bound is tight for single-bit statistics algorithms.

A(A/T)TGGCGA TTCCGGAG (A/G)AGGATACAT.

Next, we examine the consensus sequence in the middle row. Clearly, the consensus is longer than the original string, which is a consequence of the deletion errors and the right-shifting process for minority symbols. Furthermore, the second block is disbalanced, as there is one more GC *symbol than allowed, but there were no tie-breaks at that particular location that can help resolve the problem. Furthermore, since the previous block of* 8 *symbols was balanced, it is reasonable to assume that the "boundary" of the second and third blocks have shifted due to alignment errors. Looking ahead for the first appearance of an* AT *symbol in the consensus, we can try to ignore all symbols between the last symbol that causes a disbalance and the first occurrence of a symbol of the correct type. Note that since we had a tie for the first symbol in the first block (*A *versus* G*), it is advisable to change the break of tie to avoid excluding one extra symbol. This leads to the following modification of the consensus string.*

AATGGCGA TTCCGGAA GGGAATCA
AATGGCGA TTCCGGAA AGGATACAT.

The last block of 9 *symbols in the consensus is obviously erroneous since there is one more* AT *symbol present then as expected and the block is of length* 9 *rather than* 8*. This issue prevents us from further updating the consensus. But as described in [124], we can proceed with recruiting new traces that have not been previously used for alignment due to possible address errors to resolve issues such as the ones encountered with the third block of symbols above.*

The example motivates the ideas to be pursued for code constructions which offer provable performance guarantees for trace reconstruction algorithms, which for the best results need to be more sophisticated than simple BMA-type methods. The key insight is to group symbols into blocks with constraints such as balanced content (or runlength constraints) such as the one described above, and ensure that the boundaries of the blocks can be determined with high probability (since we saw in the example that imbalances may be indicative of boundary shifts). An additional layer of protection can be added to correct errors in the blocks whenever the deletion probability is sufficiently high. The construction, as well as the main results for coded trace reconstruction, are formally described next.

Given a code $\mathcal{C} \subseteq \{0,1\}^n$, we say that $\mathcal{C}$ can be *efficiently reconstructed from* $t(n)$ *traces* if there exists a polynomial $p(n) = \Omega(n)$ and a polynomial-time algorithm $\mathcal{R}$ such that for every $\mathbf{c} \in \mathcal{C}$ one has

$$P_{T_1,\ldots,T_t}[\mathcal{R}(T_1,\ldots,T_t) = \mathbf{c}] \geqslant 1 - 1/p(n),$$

where the traces T_i, $i \in [t]$, are i.i.d. according to the output distribution of the deletion channel with deletion probability δ when presented with the input $\mathbf{c}$. This definition corresponds to the worst-case trace reconstruction problem restricted to codestrings of $\mathcal{C}$. The goal of coded trace reconstruction is to design *efficiently encodable* codes $\mathcal{C}$ that can be efficiently reconstructed from $t(n)$ traces for $t(n)$ as small as possible. As a remark, we require a reconstruction success probability $1-1/p(n)$ in order to be able to compare the results of coded reconstruction with those of unrestricted trace reconstruction.

At a high level, the simplest construction splits an n-bit message into shorter blocks of length $O(\log^2 n)$, encodes each block with an inner code satisfying a certain constraint (such as a runlength/balaning and/or more general error-correcting constraints), and adds markers of length $O(\log n)$ between the blocks. Markers are of the form $0^{c\log n}1^{c\log n}$, where c is a constant, i.e., markers are concatenations of sufficiently long runs of 0s and 1s that are prohibited from occurring within the blocks. The structure of the markers and the property of the code used for the blocks ensure that with high probability, one can split the traces into shorter blocks associated with substrings of length $O(\log^2 n)$, and then run some worst-case trace reconstruction algorithm on the blocks individually. As a result, for every constant deletion probability $\delta < 1$, one can ensure the existence of an efficiently encodable code $\mathcal{C} \subseteq \{0,1\}^{n+r}$ with redundancy $r = O(n/\log n)$ that can be efficiently reconstructed from $\exp(O(\log^{2/3} n))$ traces. Note that reconstruction only requires identifying the markers and reconstructing (in parallel) multiple short-length blocks.

This construction can be further improved while preserving the efficiency of encoding and reconstruction by repeating the process, i.e., making the approach nested. More precisely, we can perform a further partition of all blocks into even shorter subblocks and add a second level of markers: each block of length $\log^2 n$ can be partitioned into blocks of length $(\log\log n)^2$, with markers of length $O(\log\log n)$ added between them. The reconstruction procedure is almost identical to the one already described, except for the fact that a small fraction of blocks will very likely not be reconstructed properly. This issue can be resolved by adding error-correction redundancy to the string to be encoded, resulting in the following claim. For every constant deletion probability $\delta < 1$, there exists an efficiently encodable code $\mathcal{C} \subseteq \{0,1\}^{n+r}$ with redundancy $r = O(n/\log\log n)$ that can be efficiently reconstructed from $\exp(O(\log\log n)^{2/3})$ traces.

Even this result can be further improved provided that the deletion probability is a sufficiently small constant, in which case modified average-case trace reconstruction algorithms can be used to substantially reduce the number of traces required. This can be achieved with a negligible rate loss. The key idea is that one can efficiently encode n-bit messages into strings that are *almost subsequence-unique* via constructions based on *almost* κ*-wise independent random variables* [3]. The enabling result for this type of trace reconstruction is the average-case algorithm from [51] which is specifically designed to operate on subsequence-unique strings.

A random vector $X \in \{0,1\}^m$ is said to be ϵ*-almost* κ*-wise independent* if for all sets of κ distinct indices $i_1, i_2, \ldots, i_\kappa \in \{1,\ldots,m\}$, one has

$$|P[X_{i_1} = x_1,\ldots,X_{i_\kappa} = x_\kappa] - 2^{-\kappa}| \leqslant \epsilon,$$

for all $(x_1,\ldots,x_\kappa) \in \{0,1\}^\kappa$. In words, we require that every possible κ-subsequence has probability close to $2^{-\kappa}$ – the probability distribution induced by any collection of κ coordinates of the random vector is close to uniform.

Interestingly, such random vectors have explicit constructions based on expander graphs or duals of BCH codes [3], [85]. The trace reconstruction algorithm that exploits the structure of almost subsequence-unique strings relies on an interesting voting strategy that *does not treat every trace equally,* but weighs them according to their reliability. Since analyzing the reconstruction method using probability measures that capture the reliability of traces is difficult, the authors of [51] suggest to only use traces that match the last $O(\log n)$ already reconstructed bits (as this gives high confidence that the trace is "synchronized" with the current estimate). Note that the idea behind this approach is somewhat reminiscent of the one used in [124], where the accuracy of traces was estimated based on the accuracy of the address strings that are known to both the information reader and writer. Furthermore, In addition to combining constructions of almost subsequence-unique strings with the corresponding average-case reconstruction algorithm, one also needs to carefully adapt the marker-based approach since the bootstrapping approach used in [51] fails for the concatenated runs case.

With these considerations in mind, one can prove the following results, stated in [24]. First, there exists an absolute constant $\delta^\star > 0$ such that for all $\delta \leqslant \delta^\star$ there exists an efficiently encodable code $\mathcal{C} \subseteq \{0,1\}^{n+r}$ with redundancy $r = O(\log n)$ that can be efficiently reconstructed from $\mathrm{poly}(n)$ traces with deletion probability δ. Second, there exists another absolute constant $\delta^\star > 0$ (to avoid notational clutter, we used the same notation although the constants are different) such that for all $\delta \leqslant \delta^\star$ there exists an efficiently encodable code $\mathcal{C} \subseteq \{0,1\}^{n+r}$ with redundancy $r = O(n/\log n)$ that can be efficiently reconstructed from $\mathrm{poly}(\log n)$ traces with deletion probability δ.

Next, we describe how to convert results pertaining to binary codes to codes over larger alphabets. The main claim is that the existence of a binary trace-reconstruction code $\mathcal{C}$ of length n with rate R that can be efficiently encoded and reconstructed from t traces with error probability ϵ implies the existence of a q-ary code $\mathcal{C}'$, where $q = 2^k$, of the same rate R. The latter can also be efficiently encoded and reconstructed from t traces with error probability at most $k\epsilon$.

To see this, consider a code whose codestrings are concatenations of binary codestrings from $\mathcal{C}$ of the form shown below.

$$\mathcal{C}' = \{(c^1, c^2, \ldots, c^k) : c^i \in \mathcal{C}, i = 1, \ldots, k\} \subseteq \{0,1\}^{k \cdot n}.$$

Clearly, the code $\mathcal{C}'$ can be viewed as a q-ary code of length n and rate R by considering an encoding of the q-ary symbols using the k binary coordinates of the strings $c^i, i \in [k]$. Next, suppose that T' is a trace of some codestring $c' = (c^1, c^2, \ldots, c^k) \in \mathcal{C}'$. Observe that the trace T_i is obtained by replacing each q-ary symbol in T' by the i-th bit of its binary expansion (which is probabilistically equivalent to a trace of c^i). As a result, applying the transformation $T \mapsto T^i$ to each of the t traces of c' and running the reconstruction algorithm associated with $\mathcal{C}$ allows us to recover c^i with error probability at most ϵ.

Since this holds for every $i = 1, \ldots, k$, a simple application of the union bound over all indices i shows that we can simultaneously recover $c^1, c^2, \ldots, c^k$ from t traces of c' with an error probability that satisfies $\leqslant k\epsilon$. If the above described constituent binary codes are efficiently encodable, then the q-ary codes are efficiently encodable as well. The codes can also be designed to ensure balanced GC-content. To satisfy the balancing constraint, one has to use different markers and a specialized code over the blocks. More precisely, within the blocks, balanced markers of the form $(AC)^\ell (TG)^\ell$ with $\ell = 25 \log n$ are used instead of the binary runlength markers, where n as before denotes the codelength.

Consequently, we have the following results for q-ary codes. For every constant deletion probability $\delta < 1$, there exists an efficiently encodable code $\mathcal{C} \subseteq \{A, C, G, T\}^{n+r}$ with redundancy $r = O(n/\log n)$ and balanced GC-content that can be efficiently reconstructed from $\exp(O(\log^{2/3} n))$ traces. For every constant deletion probability $\delta < 1$, there exists an efficiently encodable code $\mathcal{C} \subseteq \{A, C, G, T\}^{n+r}$ with redundancy $r = O(n/\log\log n)$ and balanced GC-content that can be efficiently reconstructed from $\exp(O(\log\log n)^{2/3})$ traces. A summary of the coded trace reconstruction results is available in Table I.

We conclude this exposition by referring the interested reader to a *hybrid coded trace reconstruction approach* [37], which in addition to traces uses combinatorial families known as k-decks, i.e., collections of all subsequences of length k of a given string of length n.

C. Set-Codes With Small Discrepancy for DNA Punchcards

Code designs for DNA Punchcards are fundamentally different from those used in other molecular storage systems. DNA Punchcards have readily available native sequences that serve as references for alignment of the fragments created via nicking. In all experiments performed on this system (which were of moderate scale), no alignment or readout errors were observed. Consequently, no error-correction was needed to ensure correct reading of the nicking information. However, this type of native DNA-based storage platform suffers from *duplex stability* issues. Stability problems arise when nicks are placed in close proximity of each other, causing disassociation of the DNA fragment straddled by the nicks. Since a nick can be placed either on the $3' - 5'$ or $5' - 3'$ strand, distributing the nicks in a nearly balanced fashion across the strands is expected to increase duplex stability. Furthermore, if the number of sites actually nicked is small compared to the total number of available nicking sites, the disassociation problem is reduced further. The only conceivable way in which an error could occur is to either have defective guides that fail to recognize the correct locations to be nicked or off-target nicking activities. Therefore, requiring further that the combinations of nicked locations of different codestrings differ substantially would resolve these issues as well.

To construct balanced and nonconfusable nick-based codestrings, we will use the notion of *set discrepancy,* introduced in [11]. Set discrepancy theory has been studied in a number of works [31], [65], [83], and has found applications in pseudorandomness and independent permutation generation [4], [100], ϵ-approximations and geometry [73], bin packing,

TABLE I

SUMMARY OF THE PROPERTIES OF TRACE RECONSTRUCTION CODES. TO AVOID NOTATIONAL CLUTTER, CONSTANTS IN THE EXPRESSIONS FOR THE REDUNDANCY AND NUMBER OF TRACES ARE OMITTED

Code redundancy	Number of traces t	Parameter values & Sequence properties
$\frac{n}{\log n}$	$\exp(\log^{2/3} n)$	Arbitrary constant deletion probability, balanced GC-content
$\frac{n}{\log\log n}$	$\exp((\log\log n)^{2/3})$	Arbitrary constant deletion probability, balanced GC-content
$\frac{n}{\log n}$	$\mathrm{poly}(\log n)$	Sufficiently small constant deletion probability

lattice approximations and graph spectral analysis [30], [97], [109].

Informally, the discrepancy of a finite family of subsets over a finite ground set equals the smallest integer d for which the elements in the ground set may be labeled by one of the labels ± 1 so that the absolute value of sums of labels within each subset is at most d (note that the notation d for discrepancy used in this section is not to be confused with the notion of minimum asymmetric distance from the previous sections). In a sense, discrepancy measures how difficult it is to find a labeling of elements that would keep all subsets of the family as close to being balanced as possible.

The formal definition for our storage problem is as follows. A family of subsets over $[n]$, $\mathcal{F}_n = \{F_1, \ldots, F_s\}$, $s \geqslant 2$, is termed k-*regular* if for all $1 \leqslant j \leqslant s$, $|F_j| = k$. Let $L : [n] \to \{+1, -1\}$ be a labeling of the elements in $[n]$. The *discrepancy* of a set $F_j \in \mathcal{F}_n$ under the labeling L is defined as $D_L(F_j) = \left|\sum_{i \in F_j} L(i)\right|$. The discrepancy of the family $\mathcal{F}_n$ of sets is defined as

$$D(\mathcal{F}_n) = \min_{L} \max_{1 \leqslant j \leqslant s} \left|\sum_{i \in F_j} L(i)\right|.$$

Although we focus on regular families $\mathcal{F}_n$, there is no inherent reason why one cannot use irregular families as well.

For the particular problem of code design for Punchcard systems, we are interested in families of sets $\mathcal{F}_n$ that have *small intersections*, since the sets in the family $\mathcal{F}_n$ are to represent "codesets" (i.e., we choose to represent codestrings as sets indicating the locations of nonzero/nicked symbols) whose every coordinate is a potential nicking site. By using codesets to represent combinations of nicking sites, it is natural to require that the codesets have small intersections (i.e., the codestrings to have largely mismatched locations of nonzero/nicked symbols). The codeset formalism also allows for simpler formulations of the coding problems in terms of set discrepancy and set intersection constraints.

Next, we say that the sets in $\mathcal{F}_n$ have b-*bounded intersections* if for all pairs of distinct integers $i, j \in [s]$, $|F_i \cap F_j| < b$. Clearly, for a k-regular family $\mathcal{F}_n$, $b < k$, since we do not allow repeated sets. For fixed values of n and b, our goal is to find the largest size of a b-bounded intersection family $\mathcal{F}_n$ for which there exists a labeling L such that $D_L(F_j) \in \{-1, 0, +1\}$ for all $1 \leqslant j \leqslant s$. We refer to such a set system as an *extremal balanced family*.

A line of work addressing a similar balanced set-family question in the context of combinatorial designs appeared in [27]. The problem studied is that of *bi-coloring* of Steiner triple systems (STSs). Roughly speaking, Steiner triple systems are set systems in which the subsets of interest satisfy intersection constraints that ensure that each pair of distinct elements of the ground set appears in exactly one subset (block) of the system. The key finding is that STSs are not perfectly bi-colorable, i.e., that there will always exist a monochromatic triple in the STS.

To design extremal balanced families, one can start with known families of sets with small intersections, such as the Bose-Bush and Babai-Frankl families [6], [13]. In this case, one can achieve the smallest possible discrepancy ($d = 0$ for even-sized sets and $d = 1$ for odd-sized sets) in a natural manner, by using only the defining properties of the sets.

Let q be a prime power such that $1 \leqslant b \leqslant k \leqslant q$, and let $n = kq$. Furthermore, let ξ be a primitive element of the finite field $\mathbb{F}_q$. Let $\mathcal{A} = \{0, 1, \xi, \ldots, \xi^{k-2}\}$, so that $|\mathcal{A}| = k$. For each polynomial $f \in \mathbb{F}_q[x]$, define a set of ordered pairs of elements from the underlying finite field according to

$$A_f \triangleq \{(a, f(a)) : a \in \mathcal{A}\}.$$

Clearly, $|A_f| = k$ since $|\mathcal{A}| = k$. Furthermore, let

$$\mathcal{C}(k, q) \triangleq \{A_f : f \in \mathbb{F}_q[x], \deg(f) \leqslant b - 1\}. \tag{7}$$

Then $\mathcal{C}(k, q)$ is a family of q^b k-subsets of the set $\mathcal{X} \triangleq \mathcal{A} \times \mathbb{F}_q$ such that every two sets intersect in at most $b - 1$ elements. This follows because two distinct polynomials of degree $\leqslant b-1$ cannot intersect in more than $b-1$ points. The Ray-Chauduri and Wilson Theorem [6] asserts that the size of any family $\mathcal{F}_n$ of k-regular sets with $k \geqslant b$ whose pairwise intersection cardinalities lie in some set of cardinality b satisfies $|\mathcal{F}_n| \leqslant \binom{n}{b}$. As an example, the set of all b-subsets of $[n]$ forms a $(b-1)$-intersection bounded b-regular family of subsets. Under certain mild parameter constraints, the aforementioned result can be strengthened when the set of allowed cardinalities equals $\{0, 1, \ldots, b-1\}$. The size of the family is roughly $\frac{n^b}{k^b}$.

Given the simple definition in (7), one can easily devise a labelling L of the pairs of points $(a, f(a))$ such that every set in the family $\mathcal{C}(k, q, s)$ has discrepancy = 0, for even k, and discrepancy = ± 1, for odd k. For completeness, we present the very simple proof of this claim from our work [35].

The first step consists in disposing of the representation of a point in terms of a pair of symbols from the underlying finite field. To this end, we use a map M that operates on $\mathbb{F}_q$ and is such that $M(0) = 0$ and $M(\alpha) = m + 1$ if $\alpha = \xi^m \neq 0$. It is easy to see that $M(\alpha) \in [0, k-1]$, $\forall\, \alpha \in \mathcal{A}$ and that $M(\beta) \in [0, q-1]$, $\forall\, \beta \in \mathbb{F}_q$. A pair $(\alpha, \beta) \in \mathcal{X} = \mathcal{A} \times \mathbb{F}_q$ is mapped to $\sigma(\alpha, \beta) = qM(\alpha) + M(\beta) \in [0, n-1]$, and M is a bijection.

Assume that k is even. Then, for every set A_f, one half of the elements are mapped to $[0, n/2 - 1]$ while the other half

of the elements are mapped to $[n/2, n-1]$. To see why this claim holds, note that for $\alpha \in \{0, 1, \xi, \ldots, \xi^{k/2-2}\} \subset \mathcal{A}$ and $\beta = f(\alpha) \in \mathbb{F}_q$, the pair (α, β) is mapped to

$$\sigma(\alpha, \beta) = qr(\alpha) + r(\beta) \leqslant q(k/2 - 1) + (q - 1) = n/2 - 1.$$

In a similar manner, for $\alpha \in \{\xi^{k/2-1}, \ldots, \xi^{k-2}\} \subset \mathcal{A}$ and $\beta = f(\alpha)$, we have

$$\sigma(\alpha, \beta) = qr(\alpha) + r(\beta) \geqslant qk/2 + 0 = n/2.$$

Based on this result, one can construct the labeling L as follows: assign -1 to (α, β) if $\sigma(\alpha, \beta) < n/2$, and assign $+1$ to (α, β) if $\sigma(\alpha, \beta) \geqslant n/2$. Then, every set in the family has half of the elements mapped to -1 and half mapped to $+1$. Equivalently, the discrepancy of every set equals 0. The case when k is odd can be handled in the same way.

Three remarks are in place. First, the balancing property directly follows from the simple partition of the set $\mathcal{A}$. Second, the construction of the sets is reminiscent of the ubiquitous Reed-Solomon construction. Third, the already mentioned connection of the coding problem to combinatorial design theory suggests other constructions; throughout the remainder of the subsection, we focus on discussing one such approach based on *transversal designs.*

A *transversal design* [26] $TD(t, k, v)$ consists of a set V of kv elements, called points, and a partition of V into sets $\{G_i : i \in [k]\}$, called groups. All groups $G_i, i \in [k]$, contain exactly v points. In addition, we have a set $\mathcal{B}$ of k-subsets called blocks. A block and a group obey intersection constraints that can be summarizes as follows: every b-subset of V is either contained in exactly one block or in exactly one group, but not both. Because no b-subset of elements can appear in two or more blocks, any two distinct blocks of a $TD(b, k, v)$ intersect in at most $b - 1$ elements. Therefore, whenever a $TD(b, k, v)$ exists, one can use it to construct a family of sets with small intersections that are simultaneously balanced by mimicking the proof described above. To summarize, we assign $+1$ labels to the points in half of the groups and -1 labels to the points in the other half of the groups when k is even (and follow a similar approach for odd k). It is straightforward to see that the Bose-Bush/Babai-Frankl construction actually represent a transversal design, which was first pointed out in [112]. Furthermore, it is not difficult to add k-blocks to the design and still retain the balancing and intersection constraints.

For simplicity, assume that k is even and that $b \geqslant 3$. There has to exist one group in the design that is properly contained within the set of positively labeled elements (which we henceforth denote by P_+), and one group that is properly contained within the set of negatively labeled elements (which we henceforth denote by P_-). A simple counting argument reveals that there are $\left(\frac{k}{2}\right)^2$ such pairs of groups. By construction, any k-subset with $\frac{k}{2}$ points from the first group and $\frac{k}{2}$ points from the second group intersects each block of the transversal design in at most two points. Furthermore, each pair of blocks of the type above intersects in at most $\lceil \frac{k}{2} \rceil$ points. Hence, if $b > \max\{\frac{k}{2}, 2\}$, the blocks used to augment the design are both balanced and satisfy the required intersection constraint. This construction easily extends to larger selections of groups: as long as $b > \max\{\frac{k}{s}, s\}$, where s is the size of the collection of groups, the new blocks satisfy the required constraints provided that any two collections of s groups share less than $\frac{b}{k/s}$ groups. An important conclusion that arises from this argument is that transversal designs and related combinatorial designs may not directly lend themselves to constructions of extremal balanced families of sets with small intersections. Instead, one may need to use combinations of designs, and several such constructions based on mutually orthogonal Latin squares and derivatives of orthogonal arrays and packings have been reported in [35]. For recent extensions of the above results, please refer to [130].

D. Coding for Unique Reconstruction

The three different coding-theoretic problems pertaining to DNA error-correction and constrained coding for shotgun and nanopore sequencing, as well as DNA Punchcard systems, do not deal with another fundamental class of problems termed *unique string reconstruction*. With the constraints imposed by individual sequencing devices on the type of outputs produced, one of the most important outcomes is to ensure that even in the ideal case of no sequencing errors, a DNA string can be uniquely reconstructed from the available output data of the sequencer.

Example 9: To illustrate this requirement, consider the following example of two distinct binary strings, $\mathbf{x} = 10010$ *and* $\mathbf{y} = 00100$. *Let* $S_\ell(\mathbf{z})$ *denote the set of all substrings of the string* $\mathbf{z}$ *of length* ℓ. *Then,* $S_3(\mathbf{x}) = S_3(\mathbf{y}) = \{100, 001, 010\}$, *and based on the substring information alone, one cannot distinguish the strings* $\mathbf{x}$ *and* $\mathbf{y}$. *Increasing* ℓ *from* 3 *to* 4 *leads to* $S_4(\mathbf{x}) = \{1001, 0010\} \neq S_4(\mathbf{y}) = \{0010, 0100\}$. *Therefore, based on the two substrings of length* 4, *one can discriminate the two possible (input) strings.*

A general result regarding uniqueness of string reconstruction based on substrings of length ℓ was derived in [118], where it was shown that a string is uniquely ℓ-substring reconstructable if all its $\ell - 1$-substrings occur at most once (i.e., if there are no repeats). Other important result in the area [72], [108] established that unique ℓ-substring reconstruction is impossible for strings with period $p \leqslant \ell$ (a string $\mathbf{x}$ is said to have period p if $x_i = x_{i+p}$, for all $1 \leqslant i \leqslant n - p$). Otherwise, $\ell \geqslant \lfloor n/2 \rfloor + 1$ suffices for unique reconstruction. For example, $S_4(0111011) = S_4(1110111) = \{0111, 1110, 1011, 1101\}$, since $p = 4$ and $\ell = 4$.

Native (mamalian) DNA usually contains a large numbers of repeats [54] and as a result, modern sequencing technologies are being redesigned to produce long reads [52] that can use the context of the repeats to ensure unique reconstruction. Adapting the sequence content for ease of reconstruction is, in this case, obviously impossible. But once again, that is not the case for DNA-based data storage applications, since one can encode the strings to avoid repeats, as first suggested in [38]. The problem addressed in [38] can be summarized as follows. Let $\mathcal{C}_\ell$ be a set of binary codestrings $\mathbf{x}$ of length n, each of which can be uniquely reconstructed based on $S_\ell(\mathbf{x})$. What is the largest size of $\mathcal{C}_\ell$ for a given ℓ and can the code(s) be efficiently encoded and decoded? The

question was addressed affirmatively, establishing that codes $\mathcal{C}_\ell$ of asymptotic rate equal to 1 (more precisely, including only a constant number of redundant bits) exist whenever the substrings are long enough, i.e., $\ell > 2\log(n)$. These codes can be encoded using a specialized *repeat-removal* procedure, which replaces repeats with pointers to the locations of their first occurrence, reminiscent of but significantly more involved than a related procedure for runlength coding [119]. Related problems and generalizations thereof are also discussed in [32] and [129].

Other relevant coding methods for unique string reconstruction include [2], [39], [90], and [40]. There, strings are reconstructed based on masses (i.e., weights) of their substrings, or prefixes and suffixes only, without knowing the actual substrings themselves. This subsequence-weight reconstruction problem is motivated by *mass spectrometry sequencing* [21] and its application to data storage in synthetic polymers [61]. The interested reader is referred to the original manuscripts for an in-depth coverage of the topics, with solutions including mixtures of ideas from the area of the turnpike reconstruction problem [28], code constructions based on Catalan strings and modifications thereof [111] and binary B_h sequences [40] and constant-weight codes [107]. Recent extensions and generalizations are available in [8], [47], and [128].

As a concluding remark, despite the superficial similarity to lossless universal compression methods such as Lempel-Ziv encoding [131], [132], the approaches used for unique reconstruction are substantially different. For example, with repeat removal, one only eliminate redundancy in the form of exactly repeated substrings of a predetermined length (or range of lengths), without trying to build a dictionary that can be used to compress the string. Furthermore, for polymer-based coding, one is allowed to only use information about weights of substrings to perform reconstruction, since polymer readouts are frequently performed via mass spectrometry analysis.

V. Open Problems

Many open coding-theoretic problems in the area remain and new arise due to constant changes and improvements in the synthesis and sequencing protocols used for DNA-based data storage. We list some of the problems below, grouped according to the four topics outlined in the previous subsections.

- *Sequence reconstruction and error-correction for paired-end DNA sequencing reads.* Current Illumina platforms allow for reading long DNA fragment from two ends simultaneously, thereby providing information about a pair of substrings as well as the *distance* between them. Paired reads can resolve issues with repeats and also help detect genomic rearrangements which are due to DNA breakage [41]. The questions of interest in this context are to repeat the analysis of DNA sequence profiles with additional distance information for the paired substrings, both in the presence of missing pairs of substrings or errors in the paired readout content.
- *Coded trace reconstruction for combinations of indel and substitution errors.* Instead of using trace reconstruction models for nanopore sequencers that solely account for deletion errors, one can use more realistic abstractions that can handle insertions as well as other types of errors [56]. In addition, the problem of coded trace reconstruction for strings that satisfy additional constrains (such as bounded maximum repeat length) remains open.
- *Coded gapped* k*-decks and trace reconstruction problems.* In [44], the authors proposed the study of *gapped* k*-deck reconstruction.* As already pointed out, the k-deck of a sequence is the multiset of all subsequences of the sequence of length k. Gapped k-decks restrict the available subsequences to not include adjacent symbols of the original sequence and they can be used to model "skip" events in the readout process. To the best of the author's knowledge, nothing is known about gapped trace reconstruction or other forms of trace reconstruction in which the deletions do not follow an i.i.d model.
- *Unique sequence reconstruction for hybrid sequencing technologies.* This problem has been discussed in a very basic setting in [37], with the goal to describe how long and short read technologies can be combined to reconstruct strings. The approach uses combinations of k-decks and long traces, but does not truly combine information provided by long traces *(subsequences)* and short reads *(substrings)*. This is challenging analytical problem whose solution can potentially lead to significant and low-cost improvements of the readout channel.

References

[1] M. Abroshan, R. Venkataramanan, L. Dolecek, and A. G. I. Fabregas, "Coding for deletion channels with multiple traces," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 1372–1376.

[2] 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.

[3] N. Alon, O. Goldreich, J. Håstad, and R. Peralta, "Simple constructions of almost k-wise independent random variables," *Random Struct. Algorithms*, vol. 3, no. 3, pp. 289–304, 1992.

[4] R. Armoni, M. Saks, A. Wigderson, and S. Zhou, "Discrepancy sets and pseudorandom generators for combinatorial rectangles," in *Proc. 37th Conf. Found. Comput. Sci.*, 1996, pp. 412–421.

[5] N. Athreya, O. Milenkovic, and J.-P. Leburton, "Interaction dynamics and site-specific electronic recognition of DNA-nicks with 2D solid-state nanopores," *Npj 2D Mater. Appl.*, vol. 4, no. 1, p. 32, Sep. 2020.

[6] L. Babai and P. Frankl, "Linear algebra methods in combinatorics: With applications to geometry and computer science," Dept. Comput. Sci., Univ. Chicag, Chicago, IL, USA, Tech. Rep., p. 216, 1992.

[7] V. Baldoni et al., "A user's guide for LattE integrale v1. 7.2," *Optimization*, vol. 22, p. 2, 2014. [Online]. Available: http://www.math.ucdavis.edu/~latte/

[8] A. Banerjee, A. Wachter-Zeh, and E. Yaakobi, "Insertion and deletion correction in polymer-based data storage," *IEEE Trans. Inf. Theory*, vol. 69, no. 7, pp. 4384–4406, Jul. 2023.

[9] A. Barvinok and J. E. Pommersheim, "An algorithmic theory of lattice points in polyhedra," *New Perspect. Algebr. Combinatorics*, vol. 38, pp. 91–147, Aug. 1999.

[10] T. Batu, S. Kannan, S. Khanna, and A. McGregor, "Reconstructing strings from random traces," in *Proc. 15th Annu. ACM-SIAM Symp. Discrete Algorithms (SODA)*, 2004, pp. 910–918.

[11] J. Beck, "Balanced two-colorings of finite sets in the square I," *Combinatorica*, vol. 1, no. 4, pp. 327–335, Dec. 1981.

[12] B. Bollobás, *Modern Graph Theory*, vol. 184. Berlin, Germany: Springer, 1998.

[13] R. C. Bose and K. A. Bush, "Orthogonal arrays of strength two and three," *Ann. Math. Statist.*, vol. 23, no. 4, pp. 508–524, Dec. 1952.

[14] J. Brakensiek, R. Li, and B. Spang, "Coded trace reconstruction in a constant number of traces," in *Proc. Annu. Symp. Found. Comput. Sci. (FOCS)*, Nov. 2020, pp. 482–493.

[15] N. G. Bruijn, "Acknowledgement of priority to C. Flye Sainte-Marie on the counting of circular arrangements of 2^n zeros and ones that show each n-letter word exactly once," Dept. Math., Eindhoven Univ. Technol., Eindhoven, The Netherlands, Tech. Rep., p. 16, 1975.

[16] C. Cao, L. Krapp, A. Ouahabi, A. Radenovic, J.-F. Lutz, and M. D. Peraro, "Decoding digital information stored in polymer by nanopore," *Biophysical J.*, vol. 120, no. 3, p. 98a, Feb. 2021.

[17] S. Chandak et al., "Overcoming high nanopore basecaller error rates for DNA storage via basecaller-decoder integration and convolutional codes," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 8822–8826.

[18] Z. Chang, J. Chrisnata, M. F. Ezerman, and H. M. Kiah, "Rates of DNA sequence profiles for practical values of read lengths," *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7166–7177, Nov. 2017.

[19] Z. Chase, "New upper bounds for trace reconstruction," 2020, *arXiv:2009.03296*.

[20] Z. Chase, "New lower bounds for trace reconstruction," *Annales de l'institut Henri Poincare (B) Probab. Statist.*, vol. 57, no. 2, pp. 627–643, 2021.

[21] C. Chen, J. Hou, J. J. Tanner, and J. Cheng, "Bioinformatics methods for mass spectrometry-based proteomics data analysis," *Int. J. Mol. Sci.*, vol. 21, no. 8, p. 2873, Apr. 2020.

[22] T. Chen, A. Solanki, and M. Riedel, "Parallel pairwise operations on data stored in DNA: Sorting, shifting, and searching," in *Proc. 27th Int. Conf. DNA Comput. Mol. Program.*, 2021, pp. 1–21.

[23] K. Cheng, E. Grigorescu, X. Li, M. Sudan, and M. Zhu, "On k-mer-based and maximum likelihood estimation algorithms for trace reconstruction," 2023, *arXiv:2308.14993*.

[24] M. Cheraghchi, R. Gabrys, O. Milenkovic, and J. Ribeiro, "Coded trace reconstruction," *IEEE Trans. Inf. Theory*, vol. 66, no. 10, pp. 6084–6103, May 2020.

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

[26] C. J. Colbourn, *CRC Handbook of Combinatorial Designs*. Boca Raton, FL, USA: CRC Press, 2010.

[27] C. J. Colbourn, J. H. Dinitz, and A. Rosa, "Bicoloring Steiner triple systems," *Electron. J. Combinatorics*, vol. 6, no. 1, p. 25, May 1999.

[28] T. Dakic, "On the turnpike problem," Ph.D. dissertation, Dept. Comput. Sci., Simon Fraser Univ. BC, Canada, 2000.

[29] A. De, R. O'Donnell, and R. A. Servedio, "Optimal mean-based algorithms for trace reconstruction," in *Proc. 49th Annu. ACM SIGACT Symp. Theory Comput.*, Jun. 2017, pp. 1047–1056.

[30] B. Doerr, "Lattice approximation and linear discrepancy of totally unimodular matrices—Extended abstract," in *Proc. SIAM-ACM Symp. Discrete Algorithms*, 2001, pp. 119–125.

[31] B. Doerr and A. Srivastav, "Multicolour discrepancies," *Combinatorics, Probab. Comput.*, vol. 12, no. 4, pp. 365–399, Jul. 2003.

[32] O. Elishco, R. Gabrys, E. Yaakobi, and M. Médard, "Repeat-free codes," *IEEE Trans. Inf. Theory*, vol. 67, no. 9, pp. 5749–5764, Sep. 2021.

[33] R. P. Feynman, "There's plenty of room at the bottom," *Eng. Sci.*, vol. 23, no. 5, pp. 22–36, 1960.

[34] E. C. Friedberg, "Dna damage and repair," *Nature*, vol. 421, no. 6921, pp. 436–440, 2003.

[35] R. Gabrys, H. S. Dau, C. J. Colbourn, and O. Milenkovic, "Set-codes with small intersections and small discrepancies," *SIAM J. Discrete Math.*, vol. 34, no. 2, pp. 1148–1171, Jan. 2020.

[36] R. Gabrys, H. M. Kiah, and O. Milenkovic, "Asymmetric Lee distance codes for DNA-based storage," *IEEE Trans. Inf. Theory*, vol. 63, no. 8, pp. 4982–4995, Aug. 2017.

[37] R. Gabrys and O. Milenkovic, "The hybrid k-deck problem: Reconstructing sequences from short and long traces," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 1306–1310.

[38] R. Gabrys and O. Milenkovic, "Unique reconstruction of coded strings from multiset substring spectra," *IEEE Trans. Inf. Theory*, vol. 65, no. 12, pp. 7682–7696, Dec. 2019.

[39] R. Gabrys, S. Pattabiraman, and O. Milenkovic, "Mass error-correction codes for polymer-based data storage," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 25–30.

[40] R. Gabrys, S. Pattabiraman, and O. Milenkovic, "Reconstruction of sets of strings from prefix/suffix compositions," *IEEE Trans. Commun.*, vol. 71, no. 1, pp. 3–12, Jan. 2023.

[41] R. Gabrys, E. Yaakobi, and O. Milenkovic, "Codes in the Damerau distance for deletion and adjacent transposition correction," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 2550–2570, Apr. 2018.

[42] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

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

[44] R. Golm, M. Nahvi, R. Gabrys, and O. Milenkovic, "The gapped k-deck problem," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2022, pp. 49–54.

[45] 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," *Angew. Chem. Int. Ed.*, vol. 54, no. 8, pp. 2552–2555, Feb. 2015.

[46] L. J. Guibas and A. M. Odlyzko, "String overlaps, pattern matching, and nontransitive games," *J. Combinat. Theory A*, vol. 30, no. 2, pp. 183–208, Mar. 1981.

[47] U. Gupta and H. Mahdavifar, "A new algebraic approach for string reconstruction from substring compositions," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2022, pp. 354–359.

[48] L. Hartung, N. Holden, and Y. Peres, "Trace reconstruction with varying deletion probabilities," in *Proc. 15th Workshop Analytic Algorithmics Combinatorics (ANALCO)*, 2018, pp. 54–61.

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

[50] N. Holden, R. Pemantle, and Y. Peres, "Subpolynomial trace reconstruction for random strings and arbitrary deletion probability," in *Proc. Conf. Learn. Theory (COLT)*, Jul. 2018, pp. 1799–1840.

[51] T. Holenstein, M. Mitzenmacher, R. Panigrahy, and U. Wieder, "Trace reconstruction with constant deletion probability and related results," in *Proc. ACM-SIAM Symp. Discrete Algorithms (SODA)*, Jan. 2008, pp. 389–398.

[52] J. Huddleston et al., "Reconstructing complex regions of genomes using long-read sequencing technology," *Genome Res.*, vol. 24, no. 4, pp. 688–696, Apr. 2014.

[53] S. Jain, F. Farnoud, M. Schwartz, and J. Bruck, "Coding for optimized writing rate in DNA storage," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 711–716.

[54] W. R. Jelinek et al., "Ubiquitous, interspersed repeated sequences in mammalian genomes," *Proc. Nat. Acad. Sci. USA*, vol. 77, no. 3, pp. 1398–1402, 1980.

[55] 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, Aug. 2012.

[56] S. Kannan and A. McGregor, "More on reconstructing strings from random traces: Insertions and deletions," in *Proc. Int. Symp. Inf. Theory*, 2005, pp. 297–301.

[57] A. Khandelwal et al., "Self-assembled microtubular electrodes for on-chip low-voltage electrophoretic manipulation of charged particles and macromolecules," *Microsyst. Nanoeng.*, vol. 8, no. 1, pp. 1–12, Feb. 2022.

[58] H. M. Kiah, G. J. Puleo, and O. Milenkovic, "Codes for DNA sequence profiles," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3125–3146, Jun. 2016.

[59] S. Kosuri and G. M. Church, "Large-scale de novo DNA synthesis: Technologies and applications," *Nature Methods*, vol. 11, no. 5, pp. 499–507, May 2014.

[60] D. Lang et al., "Comparison of the two up-to-date sequencing technologies for genome assembly: HiFi reads of Pacific Biosciences Sequel II system and ultralong reads of Oxford Nanopore," *GigaScience*, vol. 9, no. 12, pp. 1–7, 2020.

[61] C. Laure, D. Karamessini, O. Milenkovic, L. Charles, and J.-F. Lutz, "Coding in 2D: Using intentional dispersity to enhance the information capacity of sequence-coded polymer barcodes," *Angew. Chem. Int. Ed.*, vol. 55, no. 36, pp. 10722–10725, 2016.

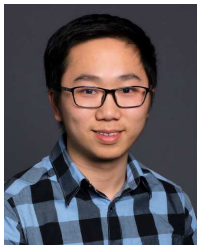
[62] A. Lenz et al., "Codes for cost-efficient DNA synthesis," in *Proc. Non-Volantile Memories Workshop (NVMW)*, 2021. [Online]. Available: http://nvmw.ucsd.edu/nvmw2021-program/nvmw2021-data/nvmw2021-paper54-presentation_slides.pdf

[63] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "The noisy drawing channel: Reliable data storage in DNA sequences," *IEEE Trans. Inf. Theory*, vol. 69, no. 5, pp. 2757–2778, May 2023.

[64] V. I. Levenshtein, "Efficient reconstruction of sequences," *IEEE Trans. Inf. Theory*, vol. 47, no. 1, pp. 2–22, Jan. 2001.
[65] L. Lovász, J. Spencer, and K. Vesztergombi, "Discrepancy of set-systems and matrices," *Eur. J. Combinatorics*, vol. 7, no. 2, pp. 151–160, Apr. 1986.
[66] J.-F. Lutz, "Coding macromolecules: Inputting information in polymers using monomer-based alphabets," *Macromolecules*, vol. 48, no. 14, pp. 4759–4767, Jul. 2015.
[67] S. Ma, I. Saaem, and J. Tian, "Error correction in gene synthesis technology," *Trends Biotechnol.*, vol. 30, no. 3, pp. 147–154, Mar. 2012.
[68] S. Ma, N. Tang, and J. Tian, "DNA synthesis, assembly and applications in synthetic biology," *Current Opinion Chem. Biol.*, vol. 16, nos. 3–4, pp. 260–267, Aug. 2012.
[69] A. Magner, J. Duda, W. Szpankowski, and A. Grama, "Fundamental bounds for sequence reconstruction from nanopore sequencers," *IEEE Trans. Mol., Biol. Multi-Scale Commun.*, vol. 2, no. 1, pp. 92–106, Jun. 2016.
[70] K. Makarychev, M. Z. Rácz, C. Rashtchian, and S. Yekhanin, "Batch optimization for DNA synthesis," *IEEE Trans. Inf. Theory*, vol. 68, no. 11, pp. 7454–7470, Nov. 2022.
[71] W. Mao, S. N. Diggavi, and S. Kannan, "Models and information-theoretic bounds for nanopore sequencing," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 3216–3236, Apr. 2018.
[72] D. Margaritis and S. S. Skiena, "Reconstructing strings from substrings in rounds," in *Proc. IEEE 36th Annu. Found. Comput. Sci.*, Oct. 1995, pp. 613–620.
[73] J. Matoušek, E. Welzl, and L. Wernisch, "Discrepancy and approximations for bounded VC-dimension," *Combinatorica*, vol. 13, no. 4, pp. 455–466, Dec. 1993.
[74] K. Mazooji and I. Shomorony, "Substring density estimation from traces," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2023, pp. 803–808.
[75] B. McBain, E. Viterbo, and J. Saunderson, "Finite-state semi-Markov channels for nanopore sequencing," 2022, *arXiv:2205.04187*.
[76] A. McGregor, E. Price, and S. Vorotnikova, "Trace reconstruction revisited," in *Proc. 22nd Annu. Eur. Symp. Algorithms (ESA)*, vol. 8737, 2014, pp. 689–700.
[77] A. S. Mikheyev and M. M. Y. Tin, "A first look at the Oxford Nanopore MinION sequencer," *Mol. Ecol. Resour.*, vol. 14, no. 6, pp. 1097–1102, Nov. 2014.
[78] O. Milenkovic, "Constrained coding for context-free languages with applications to genetic sequence modelling," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2007, pp. 1686–1690.
[79] O. Milenkovic and N. Kashyap, "On the design of codes for DNA computing," in *Proc. Int. Workshop Coding Cryptogr.* Berlin, Germany: Springer, 2006, pp. 100–119.
[80] K. Minton, "DNA typewriter," *Nature Rev. Genet.*, vol. 23, no. 9, p. 521, Sep. 2022.
[81] M. Mitzenmacher, "Capacity bounds for sticky channels," *IEEE Trans. Inf. Theory*, vol. 54, no. 1, pp. 72–77, Jan. 2008.
[82] H. Morita, A. J. van Wijngaarden, and A. J. Han Vinck, "On the construction of maximal prefix-synchronized codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 2158–2166, 1996.
[83] S. Muthukrishnan and A. Nikolov, "Optimal private halfspace counting via discrepancy," in *Proc. 44th Annu. ACM Symp. Theory Comput.*, May 2012, pp. 1285–1292.
[84] N. Nagarajan and M. Pop, "Sequence assembly demystified," *Nature Rev. Genet.*, vol. 14, no. 3, pp. 157–167, Mar. 2013.
[85] J. Naor and M. Naor, "Small-bias probability spaces: Efficient constructions and applications," in *Proc. 22nd Annu. ACM Symp. Theory Comput.*, 1990, pp. 213–223.
[86] F. Nazarov and Y. Peres, "Trace reconstruction with $\exp(O(n^{1/3}))$ samples," in *Proc. 49th Annu. ACM SIGACT Symp. Theory Comput.*, Jun. 2017, pp. 1042–1046.
[87] A. Orlitsky and S. S. Venkatesh, "On edge-colored interior planar graphs on a circle and the expected number of RNA secondary structures," *Discrete Appl. Math.*, vol. 64, no. 2, pp. 151–178, Jan. 1996.
[88] R. Oste and J. Van der Jeugt, "Motzkin paths, Motzkin polynomials and recurrence relations," *Electron. J. Combinatorics*, vol. 22, no. 2, pp. 2–8, Apr. 2015.
[89] C. Pan, S. K. Tabatabaei, S. M. H. Tabatabaei Yazdi, A. G. Hernandez, C. M. Schroeder, and O. Milenkovic, "Rewritable two-dimensional DNA-based data storage with machine learning reconstruction," *Nature Commun.*, vol. 13, no. 1, pp. 1–12, May 2022.
[90] S. Pattabiraman, R. Gabrys, and O. Milenkovic, "Coding for polymer-based data storage," *IEEE Trans. Inf. Theory*, vol. 69, no. 8, pp. 4812–4836, Aug. 2023.
[91] Y. Peres and A. Zhai, "Average-case reconstruction for the deletion channel: Subpolynomially many traces suffice," in *Proc. IEEE 58th Annu. Symp. Found. Comput. Sci. (FOCS)*, Oct. 2017, pp. 228–239.
[92] G. Pick, "Geometrisches zur zahlenlehre," *Sitzenber. Lotos (Prague)*, vol. 19, pp. 311–319, Jan. 1899.
[93] L. Qian and E. Winfree, "Scaling up digital circuit computation with DNA strand displacement cascades," *Science*, vol. 332, no. 6034, pp. 1196–1201, Jun. 2011.
[94] L. Qian, E. Winfree, and J. Bruck, "Neural network computation with DNA strand displacement cascades," *Nature*, vol. 475, pp. 368–372, Jul. 2011.
[95] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, Jun. 1960.
[96] M. G. Ross et al., "Characterizing and measuring bias in sequence data," *Genome Biol.*, vol. 14, no. 5, p. R51, 2013.
[97] T. Rothvoss, "Approximating bin packing within O(logOPT·loglogOPT) bins," in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci.*, Oct. 2013, pp. 20–29.
[98] F. Ruskey, J. Sawada, and A. Williams, "De Bruijn sequences for fixed-weight binary strings," *SIAM J. Discrete Math.*, vol. 26, no. 2, pp. 605–617, Jan. 2012.
[99] T. H. Saey, "Story one: Ancient horse's DNA fills in picture of equine evolution: A 700,000-year-old fossil proves astoundingly well preserved," *Sci. News*, vol. 184, no. 2, pp. 5–6, Jul. 2013.
[100] M. Saks, A. Srinivasan, S. Zhou, and D. Zuckerman, "Low discrepancy sets yield approximate min-wise independent permutation families," *Inf. Process. Lett.*, vol. 73, nos. 1–2, pp. 29–32, Jan. 2000.
[101] M. Schwarz, M. Welzel, T. Kabdullayeva, A. Becker, B. Freisleben, and D. Heider, "MESA: Automated assessment of synthetic DNA fragments and simulation of DNA synthesis, storage, sequencing and PCR errors," *Bioinformatics*, vol. 36, no. 11, pp. 3322–3326, Jun. 2020.
[102] N. C. Seeman, "An overview of structural DNA nanotechnology," *Mol. Biotechnol.*, vol. 37, no. 3, pp. 246–257, Oct. 2007.
[103] J. Shendure and E. L. Aiden, "The expanding scope of DNA sequencing," *Nature Biotechnol.*, vol. 30, no. 11, pp. 1084–1094, Nov. 2012.
[104] S. L. Shipman, J. Nivala, J. D. Macklis, and G. M. Church, "CRISPR–Cas encoding of a digital movie into the genomes of a population of living bacteria," *Nature*, vol. 547, no. 7663, pp. 345–349, Jul. 2017.
[105] I. Shomorony and R. Heckel, "Information-theoretic foundations of DNA data storage," *Found. Trends Commun. Inf. Theory*, vol. 19, no. 1, pp. 1–106, 2022.
[106] F. Sievers and D. G. Higgins, "Clustal omega, accurate alignment of very large numbers of sequences," in *Multiple Sequence Alignment Method*, 2014, pp. 105–116.
[107] J. Sima, Y.-H. Li, I. Shomorony, and O. Milenkovic, "On constant-weight binary B_2-sequences," 2023, *arXiv:2303.12990*.
[108] S. S. Skiena and G. Sundaram, "Reconstructing strings from substrings," *J. Comput. Biol.*, vol. 2, no. 2, pp. 333–353, Jan. 1995.
[109] J. Solymosi, "Incidences and the spectra of graphs," in *Combinatorial Number Theory and Additive Group Theory*, 2009, pp. 299–314.
[110] S. R. Srinivasavaradhan, S. Gopi, H. D. Pfister, and S. Yekhanin, "Trellis BMA: Coded trace reconstruction on IDS channels for DNA storage," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 2453–2458.
[111] R. P. Stanley, "Exercises on Catalan and related numbers," *Enumerative Combinatorics*, vol. 2, pp. 221–247, Jan. 1999.
[112] D. R. Stinson, "A general construction for group-divisible designs," *Discrete Math.*, vol. 33, no. 1, pp. 89–94, 1981.
[113] N. Stoler and A. Nekrutenko, "Sequencing error profiles of Illumina sequencing instruments," *NAR Genomics Bioinf.*, vol. 3, no. 1, Jan. 2021, Art. no. lqab019.
[114] S. K. Tabatabaei et al., "Expanding the molecular alphabet of DNA-based data storage systems with neural network nanopore readout processing," *Nano Lett.*, vol. 22, no. 5, pp. 1905–1914, Mar. 2022.
[115] S. K. Tabatabaei et al., "DNA punch cards for storing data on native DNA sequences via enzymatic nicking," *Nature Commun.*, vol. 11, no. 1, pp. 1–10, Apr. 2020.
[116] C. Thachuk, E. Winfree, and D. Soloveichik, "Leakless DNA strand displacement systems," in *Proc. 21st Int. Conf. DNA Comput. Mol. Program.*, vol. 9211. Springer-Verlag, 2015, pp. 133–153.
[117] J. Tian, K. Ma, and I. Saaem, "Advancing high-throughput gene synthesis technology," *Mol. BioSystems*, vol. 5, no. 7, pp. 714–722, 2009.

[118] E. Ukkonen, "Approximate string-matching with q-grams and maximal matches," *Theor. Comput. Sci.*, vol. 92, no. 1, pp. 191–211, Jan. 1992.
[119] A. Van Wijngaarden and K. Schouhamer Immink, "Construction of maximum run-length limited codes using sequence replacement techniques," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 200–207, Feb. 2010.
[120] R. Varshamov, "A class of codes for asymmetric channels and a problem from the additive theory of numbers," *IEEE Trans. Inf. Theory*, vol. IT-19, no. 1, pp. 92–95, Jan. 1973.
[121] B. Wang, C. Thachuk, A. D. Ellington, E. Winfree, and D. Soloveichik, "Effective design principles for leakless strand displacement systems," *Proc. Nat. Acad. Sci. USA*, vol. 115, no. 52, pp. 12182–12191, Dec. 2018.
[122] B. Wang, S. S. Wang, C. Chalk, A. D. Ellington, and D. Soloveichik, "Parallel molecular computation on digital data stored in DNA," *Proc. Nat. Acad. Sci. USA*, vol. 120, no. 37, pp. 1–10, 2023.
[123] R. R. Wick, L. M. Judd, and K. E. Holt, "Performance of neural network basecalling tools for Oxford Nanopore sequencing," *Genome Biol.*, vol. 20, pp. 1–10, Dec. 2019.
[124] S. M. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Sci. Rep.*, vol. 7, no. 1, pp. 1–6, Jul. 2017.
[125] S. M. H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Sci. Rep.*, vol. 5, no. 1, pp. 1–10, 2015.
[126] 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 Trans. Mol., Biol. Multi-Scale Commun.*, vol. 1, no. 3, pp. 230–248, Sep. 2015.
[127] S. T. Yazdi, H. M. Kiah, R. Gabrys, and O. Milenkovic, "Mutually uncorrelated primers for DNA-based data storage," *IEEE Trans. Inf. Theory*, vol. 64, no. 9, pp. 6283–6296, Sep. 2018.
[128] Z. Ye and O. Elishco, "Reconstruction of a single string from a part of its composition multiset," *IEEE Trans. Inf. Theory*, early access, Sep. 15, 2023, doi: 10.1109/TIT.2023.3315784.
[129] Y. Yehezkeally, D. Bar-Lev, S. Marcovich, and E. Yaakobi, "Generalized unique reconstruction from substrings," *IEEE Trans. Inf. Theory*, vol. 69, no. 9, pp. 5648–5659, Sep. 2023.
[130] W. Yu, Y. Xi, X. Wei, and G. Ge, "Balanced set codes with small intersections," *IEEE Trans. Inf. Theory*, vol. 69, no. 1, pp. 147–156, Jan. 2023.
[131] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inf. Theory*, vol. IT-23, no. 3, pp. 337–343, May 1977.
[132] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inf. Theory*, vol. IT-24, no. 5, pp. 530–536, Sep. 1978.

Olgica Milenkovic (Fellow, IEEE) received the master's degree in mathematics and the Ph.D. degree in electrical engineering from the University of Michigan, Ann Arbor, in 2001 and 2002, respectively. She is currently the Franklin W. Woeltge Professor in electrical and computer engineering with the University of Illinois at Urbana–Champaign (UIUC) and a Research Professor with the Coordinated Science Laboratory. She is also the Co-Founder of the Center for Artificial Intelligence and Modeling, Institute of Genomic Biology, UIUC. She heads a group focused on addressing unique interdisciplinary research challenges spanning the areas of algorithm design and computing, bioinformatics, coding theory, machine learning, and signal processing. Her scholarly contributions have been recognized by multiple awards, including the NSF Faculty Early Career Development (CAREER) Award, the DARPA Young Faculty Award, the Dean's Excellence in Research Award, and several best paper awards. In 2013, she was elected as a UIUC Center for Advanced Study Associate and a Willett Scholar, while in 2015, she was elected as a Distinguished Lecturer of the Information Theory Society. She has served as an Associate Editor for IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE TRANSACTIONS ON SIGNAL PROCESSING, IEEE TRANSACTIONS ON INFORMATION THEORY, and IEEE TRANSACTIONS ON MOLECULAR, BIOLOGICAL AND MULTI-SCALE COMMUNICATIONS. In 2009, she was the Guest-Editor-in-Chief of a special issue of IEEE TRANSACTIONS ON INFORMATION THEORY on Molecular Biology and Neuroscience, while in 2020, she was the Guest-Editor-in-Chief of a special issue of IEEE TRANSACTIONS ON INFORMATION THEORY In Memory of V. I. Levenshtein.



Chao Pan received the B.S. degree from Tsinghua University in 2017, and the M.S. and Ph.D. degrees from the University of Illinois Urbana-Champaign, in 2019 and 2022, respectively. He was a Post-Doctoral Research Associate with the Department of Electrical and Computer Engineering during the preparation of the manuscript. His research interests include applied machine learning, such as geometric deep learning and graph neural networks, and their applications in biology, such as learning-based DNA data storage systems.