Channel Model with Memory for DNA Data Storage with Nanopore Sequencing

Belaid Hamoum[1], Elsa Dupraz[2], Laura Conde-Canencia[1], Dominique Lavenier[3]
[1] Lab-STICC, CNRS UMR 6285, Université Bretagne-Sud, Lorient, France
[2] IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238, France.
[3] Univ. Rennes, CNRS-IRISA, Inria, Rennes, France

***Abstract*—This paper is dedicated to channel modeling and error-correction coding for DNA data storage with nanopore sequencing. We first propose a novel statistical model for DNA storage, which takes into account the memory within DNA storage error events, and follows the way nanopore sequencing works. Compared to existing channel models, the proposed model represents more accurate experimental datasets. We also propose a full error-correction scheme for DNA storage, based on a consensus algorithm and non-binary LDPC codes. Especially, we introduce a novel synchronization method which allows to eliminate remaining deletion errors after the consensus, before applying a belief-propagation LDPC decoding algorithm to correct substitution errors. This method exploits the LDPC code structure to correct deletions, and does not require adding any extra redundancy.**

I. Introduction

Over the last decade, the amount of generated data has been growing up exponentially, and [1] predicted that data storage needs would grow from 45 zetabytes in 2019 to 175 zetabytes by 2025. For this reason, it is necessary to find alternatives to classical storage methods (tapes, HDD, SSD, etc.). Among these, DNA data storage [2] appears as a promising solution that benefits from highly increased density and durability compared to existing storage solutions. Over the last years, improvements in DNA synthesis and sequencing techniques have made this technology more affordable, and an automated end-to-end DNA data storage device demonstrator was reported in [3]. This demonstrator includes the MinION, i.e., third-generation single molecule sequencing from Oxford Nanopore Technologies (ONT). This device offers a portable, real time, low-cost, and long-reads sequencing [4].

However, up-to-date DNA data storage systems remain very sensitive to errors introduced during synthesis and sequencing [5]. These two operations introduce not only substitution errors, like conventional memory storage systems, but also insertions and deletions at a non-negligible rate. Conventional decoding algorithms of existing error-correction solutions such as Turbo codes or Low Density Parity Check (LDPC) codes are not sufficient for DNA storage, because initially designed to efficiently protect sequences against substitution errors only.

In order to design efficient error-correction solutions for DNA storage, we first need to develop accurate statistical channel models that allow for error-correction code design and simulation, without having to use very costly experimental processes. DNA storage channel models currently used for error-correction performance evaluation either assume non-realistic independent and identically distributed (i.i.d.) errors [6], or rely on Deep-Learning approaches [7], which makes them difficult to interpret.

This paper introduces on a novel statistical DNA data storage channel model, based on a Markov chain with memory of order k, that models the MinION sequencing technology. Since this model consists of explicit probability terms, its assumptions (memory length, dependency to input sequence, etc.) are explicit as well. Further, once inferred on experimental data, this model can be easily interpreted (for instance, to identify which patterns in the input sequence are the most sensitive to specific types of errors). We show from numerical results that our model represents more accurately the experimental channel compared to the state of the art [6], [7].

The paper also addresses the design of efficient error-correction solutions for DNA storage. Apart from substitutions, most existing practical error-correction codes can correct only insertions [8], or only deletions [8], [9]. Existing solutions [10] which can efficiently correct the three types of errors have a very high complexity, except for the i.i.d. channel model which is not realistic in the DNA storage application. To fully reconstruct the original sequence, a commonly employed solution consists in a consensus/trace-reconstruction algorithm followed by standard channel decoding to correct remaining residual errors [11]–[13]. However, because a few amount of insertions and deletion are likely to remain after the consensus, the reconstruction process may fail. In order to correct the remaining errors, it was proposed in [12] to augment a conventional channel coding solution with periodical markers. These markers allow to correct insertions and deletions by re-synchronizing the output sequence with the original one, while the channel code takes in charge the substitution errors.

In this work, we consider a consensus algorithm together with Non-Binary (NB) LDPC codes that fit the quaternary DNA alphabet. As an alternative to [12] in order to handle the few remaining insertions and deletions, we propose a novel intermediate step between the consensus and the channel decoding. This intermediate step consists in re-synchronizing the sequence at the output of the consensus, by relying on the LDPC code structure rather than on additional markers. This

This work was supported by the Labex Cominlabs with funding from the French National Research Agency (ANR-10-LABX-07-01). We also acknowledge Emeline Roux for kindly providing the experimental datasets used in this work.

represents an interesting gain in terms of coding redundancy, which may allow to reduce the expensive synthesis costs.

Section II describes the DNA data storage workflow. Section III presents existing DNA storage channel models, and Section IV introduces our channel model with memory. Section V presents the proposed scheme for error-correction. Section VI is dedicated to simulation results.

II. DNA DATA STORAGE WORKFLOW

A. Synthesis

As described in [14], DNA synthesis consists of transforming digital data into experimental DNA strands, where each DNA strand is formed by a series of nucleotides 'A', 'C', 'G' and 'T'. Different synthesis techniques exist (Chemical, Enzymatic, Bacteria-Based, etc.), each exhibiting different constraints, in terms of *e.g.*, sequences length, costs, etc. Data sequences used in this work were synthesized using the chemical technique, in which oligonucleotides (short DNA molecules) are synthesized and then assembled to form the ordered sequences of nucleotides. After the synthesis, we get not one but thousands of copies of the synthesized sequence.

B. Nanopore sequencing

Sequencing is the physical process where DNA strands are read. In this work, we consider the ONT MinION [4]. In this sequencing technique, strands go through a nanopore nucleotide by nucleotide, and an electrical signal (current intensity) is sequentially output for every group of k successive nucleotides, called k-mers. Then, a utility software called the basecaller translates the current signal at the output of the sequencer into the k nucleotides corresponding to each k-mer. Multiple basecallers exist (Guppy, Albacore, Flappie,..), and they use either event segmentation or Deep-Learning approaches [15]. We considered Guppy for this work as it is the official basecaller provided by ONT. The basecaller generates fastQ files that contain the read sequences as well as related metadata.

III. EXISTING DNA DATA STORAGE CHANNEL MODELS

One major drawback of DNA data storage resides in the fact that both synthesis and sequencing introduce errors (insertions, deletions, substitutions) [5]. From an information-theoretic perspective, all the previous steps (synthesis, sequencing, basecalling) can be modeled as a channel, which takes as input a sequence composed of bases 'A', 'C', 'G', 'T', and outputs N edited replicas of the original sequence. In this section we introduce some notation and describe state-of-the-art channel models for DNA storage.

A. Notation

In what follows, we use Ins, Del, Sub, as abbreviations for Insertion, Deletion, Substitution, respectively. In addition, Match stands for "no error". In our model, $\mathbf{X}$ is the channel input sequence of length I, in which $X_i \in \{A, C, G, T\}$ is the base at position i. Then, $k\mathrm{mer}_i = (X_{i-k+1} \dots X_{i-1} X_i)$ is the $k-\mathrm{mer}$ of length k at position i. And $\mathbf{Y}$ is a sequence of length I of channel events, where $Y_i \in \{\mathrm{Ins}, \mathrm{Del}, \mathrm{Sub}, \mathrm{Match}\}$ is the event at position i. We consider that a given DNA storage channel model is defined by a set of probability distributions for the successive Y_i. For instance, $\mathbb{P}(Y_i)$ is the marginal probability of Y_i, and $\mathbb{P}(Y_i | k\mathrm{mer}_i, Y_{i-1})$ is the conditional probability of Y_i with respect to $k\mathrm{mer}_i$ and the previous event Y_{i-1}. Finally, we use $\mathbf{Z}$ to denote edited sequences at the output of the basecaller.

B. Independent and identically distributed channel model

In the literature, $\mathbf{Y}$ is often considered to be independent and identically distributed (i.i.d). Events Y_i are further assumed to be independent from the X_i, and the probabilities $\mathbb{P}(Y_i)$ are either inferred from sets of experimental data, or fixed arbitrarily for simulation purposes only [6], [12]. However, although it can simulate the correct amount of errors, this model cannot represent bursts of errors, since it assumes that Y_i is independent from Y_{i-1}. In addition, it does not take into account the effect of $k\mathrm{mer}_i$ onto error event Y_i, and we know from several other works [15] that a statistical relation between those exists.

C. DeepSimulator

DeepSimulator is another popular tool to simulate the DNA data storage system [7], [16]. DeepSimulator relies on a Deep-Learning approach combined with a basecaller. In a first Deep-Learning-based step, DeepSimulator takes as input a sequence of bases, and outputs electrical current levels after the sequencing. In a second step, the current levels are sent to a basecaller which transcripts the current levels into bases. Event sequences $\mathbf{Y}$ generated by DeepSimulator contain some memory and are statistically dependent from the input sequences $\mathbf{X}$. However, after having performed a significant amount of simulations we could observe that DeepSimulator does not always reflect well the physical DNA storage channel. For instance, as shown in Figure 1, the same kind of error appears in most of the simulated sequences in the same particular position, which does not correspond to an experimental channel model. Moreover, we could also observe an inaccurate predominance of substitutions and insertions over deletions. This shows the need to develop more accurate channel models for code design and performance evaluation.

IV. CHANNEL MODEL WITH MEMORY FOR DNA DATA STORAGE

We now introduce a novel statistical channel model for DNA data storage, that aims at solving the drawbacks just described of the existing models. The proposed model takes into account the statistical dependencies between the event sequence $\mathbf{Y}$ and the input sequence $\mathbf{X}$. It also considers that event Y_i depends on $k\mathrm{mer}_i$, which allows to model editions due to successive $k-\mathrm{mer}$ reads, according to the way the MinION sequencer works. Our model also assumes some internal memory in $\mathbf{Y}$, by considering that previous event Y_{i-1} can affect current event Y_i. This allows to take burst errors into account.

A. Probability terms

In this section, we use $B \in \{A, C, G, T\}$ to denote the obtained base after a substitution. Our model is then described by the following conditional probability distribution:

- $\mathbb{P}(Y_i|k\text{mer}_i, Y_{i-1})$ characterizes the dependency between the current event Y_i, the currently read $k\text{mer}_i$, and the previous event Y_{i-1}. This captures the probability of different types of events $Y_i \in \{\text{Ins}, \text{Del}, \text{Sub}, \text{Match}\}$ depending on $k\text{mer}_i$, and allows to consider bursts of errors through the dependency to Y_{i-1}.
- $\mathbb{P}(L|k\text{mer}_i, Y_i = \text{Ins})$ characterizes the insertion length L depending on the currently read $k\text{mer}_i$. Since only insertions can be of length $L > 1$, the probability of L is conditioned to event $Y_i = \text{Ins}$.
- $\mathbb{P}(B|k\text{mer}_i, Y_i = \text{Sub})$ characterizes the probability to substitute the last base X_i of $k\text{mer}_i$ by the base B, given that $Y_i = \text{Sub}$.

We assume that the probabilities $\mathbb{P}(Y_i|k\text{mer}_i, Y_{i-1})$ and $\mathbb{P}(L|k\text{mer}_i, Y_i = \text{Ins})$ do not vary with $k < i < I$. On the contrary, we observed from experimental data that the probability to get an error is higher at the first position $i = 1$ and at the last one $i = I$, compared with middle positions $1 < i < I$. Therefore, we allow for different probability distributions $\mathbb{P}(Y_1)$, $\mathbb{P}(\text{L}|Y_1 = \text{Ins})$ and $\mathbb{P}(Y_I)$, $\mathbb{P}(\text{L}|Y_I = \text{Ins})$ to be used when $i = 1$ and $i = I$. Finally, for $1 < i \leq k$, since no complete k-mer was observed already, we consider probabilities $P(Y_i|X_i)$, $P(B|X_i, Y_i = \text{Sub})$ and $P(L|X_i, Y_i = \text{Ins})$ that only depend on the base value of X_i.

B. Training

To train our model, i.e, to estimate all the considered probability terms, we used $V = 9$ sets of experimental data which went through the whole DNA data storage process, as described in Section II. Each of these sets provided one reference sequence taken as channel input $\mathbf{X}$, and N sequences obtained after sequencing, taken as channel outputs $\mathbf{Z}$. Then, the conditional probabilities $\mathbb{P}(Y_i = D|k\text{mer}_i, Y_{i-1} = E)$ were estimated by counting over the NV pairs $(\mathbf{X}, \mathbf{Y})$ the number of outcomes of each event $D \in \{\text{Ins}, \text{Del}, \text{Sub}, \text{Match}\}$, divided by the number of outcomes of the considered $(k\text{mer}_i, Y_{i-1} = E)$. The other probability terms were estimated following the same approach. Obtaining experimental data is very costly, while there are 4^{k+1} different combinations $(k\text{mer}_i, E)$, which grows fastly with k. In our case, when a given combination $(k\text{mer}_i, Y_{i-1} = E)$ was left unobserved, we estimated the corresponding probabilities by averaging over all observed combinations. At the end, note that the considered training is not only specific to a given value k, but also to a particular DNA data storage process, with fixed synthesis and sequencing techniques. In particular, considering updated or other synthesis and sequencing techniques requires to retrain the model from new sets of experimental data.

C. Channel simulator

Once we get all the probabilities from training, we can build a simulator that takes as input a given sequence and generates outputs according to the channel model. This model simulates the whole DNA data storage process and has three main advantages. First, in case of technology evolution (synthesis, sequencing, basecalling,..), the model can be retrained with new sets of experimental data (adaptability). Second, since our model also takes into account the basecaller, it is faster than DeepSimulator which requires to run the basecaller during simulations (faster simulations). Third, as opposed to a black-box approach, all our probabilities terms are explicit. The values of these probabilities can then be used, either to better understand how errors are introduced during the DNA storage process, or to incorporate them to any source and channel coding methods which would be considered in the workflow (explainability).

V. Full reconstruction solution

We now propose an error-correction scheme to correct insertion, deletion and substitution errors after the sequencing step. This scheme relies on three components: a consensus algorithm, a resynchronization step, and a NB-LDPC decoder. The first one aims to provide a good quality sequence based on the sequencing data redundancy, while the last two ones aim to correct residual errors. The consensus algorithm relies on the fact that known primers (short sequence of about 40 nucleotides) are added at the beginning and at the end of the sequence $\mathbf{X}$. These primers consist of sequences of known bases which are mainly used to select sequences of interest through biotechnology manipulations.

A. Consensus algorithm

The consensus algorithm we consider, called CCSA, has been specifically designed for the DNA data storage system introduced in Section II. CCSA takes as input m sequences $(\mathbf{Z}_1, \cdots, \mathbf{Z}_m)$ selected randomly from the set of N sequences output by the basecaller. Then, for given parameters T and L, it forms a directed graph whose nodes are given by the subsequences of length L that appear at least T times among the m input sequences. There is an edge between two nodes of the graph if the corresponding two subsequences overlap by at least d bases. Finally, a Viterbi-like algorithm is applied over the directed graph in order to select path between the start primer (p_1) and the end primer (p_2) with the highest score. Note that CCSA may output either one or several sequences of equal (highest) score of different lengths J close to I. The score is calculated from the nodes weights (number of occurrence of the subsequence over the m sequences) and edges weights (L minus overlap length between the two subsequences) over the path. A detailed description of the CCSA algorithm can be found in [17]. CCSA results reported in [17], as well as our own simulations, show that this algorithm is able to correct most of the errors introduced by the DNA storage process, although a few residual insertions, deletions, and substitutions remain. This is why an additional correction step based on NB-LDPC decoders is needed.

B. NB-LDPC Codes and decoders

LDPC codes are linear capacity-approaching blocks codes commonly used in communication systems to correct substitution errors from the channel. NB-LDPC codes are defined over Galois fields GF(q) of length $q \geq 2$ [18]. In this work, we use NB-LDPC codes in GF(4) for consistency with the quaternary bases alphabet. The parity check matrix H of the code is sparse, and its non-zero elements take values in GF(q). Then, any codeword $\mathbf{X}$ of the code verifies $H\mathbf{X} = 0$. The standard Belief-Propagation (BP) LDPC decoder [19] takes as input the consensus sequence $\tilde{\mathbf{X}}$ and seeks to outputs a sequence $\hat{\mathbf{X}}$ close to $\tilde{\mathbf{X}}$ and that verifies the condition $H\hat{\mathbf{X}} = 0$. However, the standard BP decoder as well as most existing LDPC decoders can only correct substitution errors. This is why we now propose a synchronization method to also correct a few amount of deletions. We only apply this synchronization step if the consensus does not output a sequence of length $J = I$, and by assuming that output sequences of length $J < I$ only result from deletions.

C. NB-LDPC codes synchronization

In this part, we define the score of a given sequence $\tilde{\mathbf{X}}$ as the number of unsatisfied parity check equations (that is the number of non-zero components in the vector $H\tilde{\mathbf{X}}$). We now describe the synchronization process when the consensus outputs a sequence $\tilde{\mathbf{X}}$ of length $I - 1$. In this process, for a given segment length l_s, we try to insert a base with arbitrary value 'A' at position 1, then at position $l_s + 1$, then $2l_s + 1$, and so on. For each considered position, we compute the corresponding score. At the end, we definitively add a base at the position $kl_s + 1$ that gives the lowest score, thus creating a new vector $\tilde{\mathbf{X}}_s$. Even if the base value is incorrect and the base position not entirely correct either, adding this base close to the correct position should greatly reduce the score by re-synchronizing the output sequence $\tilde{\mathbf{X}}_s$ with the original codeword $\mathbf{X}$. Now, if the sequence $\tilde{\mathbf{X}}$ is of lower length $I - v$, v bases are inserted one after each other in a greedy maneer: the first base is inserted at the position which gives the lowest score, then the process is repeated so as to insert the second base, and so on. This process allows to replace deletion errors by substitution errors which can then be corrected by the LDPC BP decoder. If after synchronization, the decoder fails (unsatisfied parity check equations remain), a new consensus will be restarted. Note that the segment length l_s plays a crucial role as it addresses a tradeoff between algorithm complexity and amount of substitution errors in the resulting sequence. Finally, this synchronization technique only relies on the LDPC code structure, and does not require any additional redundancy, unlike in a solution with periodical markers [12].

D. Full reconstruction solution

We now describe our full reconstruction solution. We first provide to the consensus algorithm m sequences at random among the M sequences $\mathbf{Z}$ that have correct primers p_1 and p_2 and length greater than I. The consensus algorithm then outputs several sequences $\tilde{\mathbf{X}}$ of length J. If the consensus outputs a sequence of length $J = I$, we set this sequence as $\tilde{\mathbf{X}}_s$. Otherwise, we pick a sequence of length $J < I$ as close as possible to I, and we apply the synchronization step described in Section V-C in order to get a sequence $\tilde{\mathbf{X}}_s$ of the correct length I. In both cases, the sequence $\tilde{\mathbf{X}}_s$ is passed through a BP decoder to correct residual substitution errors. If the sequence $\hat{\mathbf{X}}$ is incorrect in the sense that $H\hat{\mathbf{X}} \neq 0$, we restart the full reconstruct process (consensus + synchronization + BP decoder), from another set of m random sequences provided to the consensus. This process allows to reconstruct the original sequence $\mathbf{X}$ with a small number of restarts, as we now evaluate through simulations.

VI. Simulation results

In this part, we first compare the numerical simulations obtained with the proposed channel model to the existing ones (i.i.d. and DeepSimulator), and with the experimental data. Our channel model was trained as described in section IV, using 9 experimental datasets called barcodes, each of them containing multiple Fastq files related to one particular reference sequence. Since the MinION sequencer reads k-mers of length 6, we fixed the channel memory length to $k = 6$, and we performed the training over 22182 output sequences. Figure 1 shows for barcode03 four edit maps (i.i.d. channel model, DeepSimulator, our model, experimental data) containing positions of deletions, insertions, and substitutions observed for 1000 output sequences after an alignment with the reference sequence. We can observe that for the i.i.d model, errors are incorrectly uniformly distributed over all the sequence, while for DeepSimulator same errors of the same type seem to always appearing at same positions. Finally, the proposed channel model with memory seems to approach the most the experimental data, although some long-run error events are not properly taken into account.

We then evaluate the performance of our full sequence reconstruction method, by using the proposed channel model with memory. To do so, we considered barcodes 01 and 03 among the nine at our disposal, and encoded the corresponding sequences with a regular $(3, 6)$ LDPC code in GF(4) of size $(500, 1000)$ and code rate $R = 1/2$, constructed from a PEG algorithm. We further set segment length to $l_s = 50$ for our synchronization method. To evaluate the proposed reconstruction method, we considered three setups: (i) consensus alone, (i) consensus + NB-LDPC decoder without re-synchronization, (iii) consensus + re-synchronization + NB-LDPC decoder. For each of these setups, we applied the successive reconstruction steps 900 times, and evaluated the proportion of perfectly recovered sequences. This metric is of interest in our setup, since the condition $H\mathbf{X} = 0$ allows to decide that the original sequence was correctly retrieved, and therefore to stop the reconstruction loop. Figure 2 shows the proportion of correctly retrieved sequences for the two considered barcodes, with respect to the number m of input sequences to the consensus. For both barcodes, as expected, the success probability increases with m, before reaching a

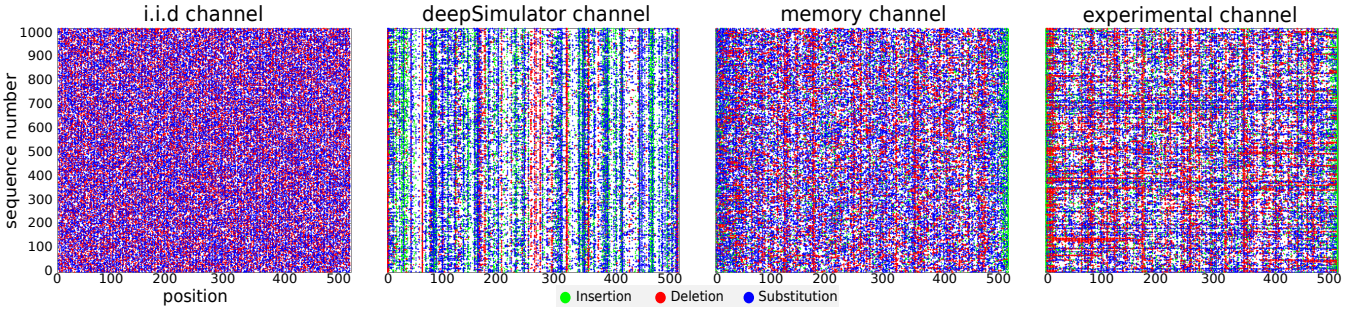

Fig. 1. Errors and types of errors observed on barcode03, for 1000 output sequences.

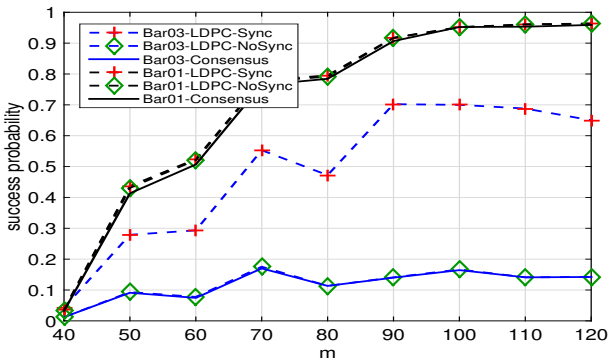

Fig. 2. Proportion of correctly retrieved sequences with respect to the number m of input sequences to the consensus.

peak at around $m = 100$. The decrease in performance after this peak probably comes from the fact that the consensus has difficulties to handle too many sequences, due to the initial majority voting operation. Then, we observe two different configurations, depending on the considered barcode. For barcode01, it seems that the resynchronization and NB-LDPC decoder are not useful, in the sense that the sequences at the output of the consensus are either correct, or very far from the original data. On the opposite, for barcode03, we observe that resynchronization greatly improves the reconstruction rate, while NB-LDPC decoder alone after consensus does not help. This is probably due to the fact that for this barcode, most sequences at the output of the consensus contain a few amount of deletions introduced by an homopolymer of length 6, which are corrected by the synchronization method.

VII. Conclusion

In this work, we proposed a channel model with memory for DNA storage. Compared to existing ones, our model represents experimental data more accurately, and should allow for efficient source/channel codes design. We then introduced a full reconstruction scheme, based on a consensus algorithm and on NB-LDPC codes, and proposed a synchronization method in order to correct remaining deletions after the consensus. Future work will include improving the channel model by considering larger sets of experimental data and optimizing the proposed error-correction scheme in terms of complexity, code rate, etc.

References

[1] D. Reinsel, J. Gantz, and J. Rydning, "The digitization of the world. from edge to core." 2018.

[2] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 7435, pp. 77–80, 2013, 23354052[pmid].

[3] C. N. Takahashi, B. H. Nguyen, K. Strauss, and L. Ceze, "Demonstration of End-to-End Automation of DNA Data Storage," *Scientific Reports*, vol. 9, no. 1, p. 4998, 2019.

[4] R. M. Leggett and M. D. Clark, "A world of opportunities with nanopore sequencing," *J. of Exp. Botany*, vol. 68, no. 20, pp. 5419–5429, 2017.

[5] R. Heckel, G. Mikutis, and R. N. Grass, "A Characterization of the DNA Data Storage Channel," *Scientific Reports*, vol. 9, no. 1, p. 9663, 2019.

[6] W. Song, K. Cai, M. Zhang, and C. Yuen, "Codes With Run-Length and GC-Content Constraints for DNA-Based Data Storage," *IEEE Communications Letters*, vol. 22, no. 10, pp. 2004–2007, 2018.

[7] S. Chandak, J. Neu, K. Tatwawadi, J. Mardia, B. Lau, M. Kubit, R. Hulett, P. Griffin, M. Wootters, T. Weissman, and H. Ji, "Overcoming High Nanopore Basecaller Error Rates for DNA Storage via Basecaller-Decoder Integration and Convolutional Codes," in *ICASSP*, 2020, pp. 8822–8826.

[8] T. Xue and F. C. M. Lau, "Construction of GC-Balanced DNA With Deletion/Insertion/Mutation Error Correction for DNA Storage System," *IEEE Access*, vol. 8, pp. 140 972–140 980, 2020.

[9] M. Abroshan, R. Venkataramanan, L. Dolecek, and A. G. i Fàbregas, "Coding for Deletion Channels with Multiple Traces," in *ISIT*, 2019, pp. 1372–1376.

[10] A. Lenz, I. Maarouf, L. Welter, A. Wachter-Zeh, E. Rosnes, and A. G. i Amat, "Concatenated Codes for Recovery From Multiple Reads of DNA Sequences," 2020.

[11] S. M. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and Error-Free DNA-Based Data Storage," *Scientific reports*, vol. 7, no. 1, pp. 5011–5011, 2017.

[12] S. Chandak, K. Tatwawadi, B. Lau, J. Mardia, M. Kubit, J. Neu, P. Griffin, M. Wootters, T. Weissman, and H. Ji, "Improved read/write cost tradeoff in DNA-based data storage using LDPC codes," in *Allerton*, 2019, pp. 147–156.

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

[14] M. Hao, J. Qiao, and H. Qi, "Current and Emerging Methods for the Synthesis of Single-Stranded DNA," *Genes*, vol. 11, no. 2, p. 116, 2020.

[15] R. R. Wick, L. M. Judd, and K. E. Holt, "Performance of neural network basecalling tools for Oxford Nanopore sequencing," *Genome Biology*, vol. 20, no. 1, p. 129, 2019.

[16] Y. Li, R. Han, C. Bi, M. Li, S. Wang, and X. Gao, "DeepSimulator: a deep simulator for Nanopore sequencing," *Bioinformatics*, vol. 34, no. 17, pp. 2899–2908, 2018.

[17] D. Lavenier, "Constrained Consensus Sequence Algorithm for DNA Archiving," *arXiv*, 2021. [Online]. Available: https://arxiv.org/abs/2105.04993

[18] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular (2, dc)-LDPC codes over GF (q) using their binary images," *IEEE Transactions on Communications*, vol. 56, no. 10, pp. 1626–1635, 2008.

[19] M. Davey and D. MacKay, "Low density parity check codes over GF(q)," in *1998 Information Theory Workshop*, 1998, pp. 70–71.