DNA-Storalator: End-to-End DNA Storage Simulator

Gadi Chaykin[1], **Nili Furman**[1], **Omer Sabary**[2], **Dvir Ben-Shabat**[1], and **Eitan Yaakobi**[1]

[1]The Henry and Marilyn Taub Faculty of Computer Science, Technion, Haifa, 3200003, Israel.
[2]Electrical and Computer Engineering Dept., University of California, San Diego, La Jolla, CA 92093, USA.

***Abstract*—DNA-Storalator is a cross-platform software tool that simulates the complete process of encoding, storing, and decoding digital data in DNA molecules. The simulator receives an input file with the designed DNA strands that store digital data and emulates the different biological and algorithmical components of the storage system. The biological component includes simulation of the synthesis, PCR, and sequencing stages which are expensive and complicated and therefore are not widely accessible to the community. These processes amplify the data and generate noisy copies of each DNA strand, where the errors are insertions, deletions, long-deletions, and substitutions. DNA-Storalator injects errors to the data based on the error rates, as they vary between different synthesis and sequencing technologies. The rates are based on comprehensive analysis of data from previous experiments but can also be customized. Additionally, the tool can analyze new datasets and characterize their error rates to build new error models for future usage in the simulator. DNA-Storalator also enables control of the amplification process and the distribution of the number of copies per designed strand. The coding components are: 1. Clustering algorithms which partition all output noisy strands into groups according to the designed strand they originated from; 2. State-of-the-art reconstruction algorithms that are invoked on each cluster to output a close/exact estimate of the designed strand; 3. Integration with external error-correcting codes and other encoding and decoding techniques. This end-to-end DNA storage simulator grants researchers from all fields an accessible complete simulator to examine new biological technologies, coding techniques, and algorithms for current and future DNA storage systems.**

I. Introduction

The emerging technology of DNA data storage has been studied extensively in the past decade due to its high capacity and stability. To store data in DNA, it is first required to design a code that defines how to encode the binary information to DNA strands. Then, using *DNA synthesis*, the DNA strands that encode the data are generated and are stored all together in a storage container. Following that, to retrieve the binary data, some synthesized DNA strands are sampled from the container and are then amplified using PCR. Then, *DNA sequencing* is performed on each of the amplified strands in order to get its symbols. Lastly the sequenced DNA strands are decoded back to binary information using the decoding procedure of the designed code. The processes of synthesis, PCR, and sequencing are all error-prone and introduce errors, while the dominant errors are insertions, deletions, and substitutions.

Due to technology limitation, the synthesized strands are limited in their length (roughly up to 300 symbols). Additionally, the current synthesis technologies cannot always generate the exact designed sequence, but a noisy version of it. Moreover, these technologies cannot generate a single copy per strand, but only multiple copies in the order of thousands to millions. As mentioned earlier, these noisy copies of each design strand are all mixed and are stored together in a storage container. Therefore, before the decoding step, it is required to group them by their original designed strands. This step is called *clustering*, and each multiset of noisy copies that were originated from the same design strand is called a *cluster*. The clustering step is usually done by *indices*, that are defined on the strands as part of the coding scheme. Each encoded strand consists of index and data (including redundancy symbols that are used for error-correction). The indices helps in the clustering step, and also defines order on the strands. After the clustering step, a *reconstruction algorithm* is performed on each cluster to estimate the designed strand. The reconstruction algorithms usually utilize the inherent redundancy of the synthesis and use the several noisy copies of each strand to correct errors. However, since they cannot always correct all of the errors, the remaining errors are corrected using an error-correcting code.

Both synthesis and sequencing technologies continue to be developed and each such a technology varies in its error distributions and characterization. Therefore, there is a crucial need to design and study coding techniques that address the different technologies, together with clustering and reconstruction algorithms. Moreover, when designing error-correcting codes for DNA storage systems, one should also address the accuracy of such algorithms on a given technology to define the required error-correction capability. In this work, we present the *DNA-Storlator*, a software tool that allows researchers from all fields to compare, study, and improve their coding techniques and algorithms with current state-of-the-art solutions. The tool can simulate the errors of the synthesis, PCR, and sequencing processes on a given design file. Then, the tool allows to its users to perform different previously published clustering and reconstruction algorithms, as well as user-designed such algorithms. Additionally, the DNA-Storalator supports the error characterization of new DNA synthesis and sequencing technologies, using a small sample of its reads. Then, users can simulate errors based on this characterization and analyze algorithms for these technologies.

II. The DNA Storalator

In this section we present the components of the DNA-Storalator. This simulator is a user-accessible tool that allows a complete simulation and algorithms analysis for all stages of DNA storage. A schematic overview of the tool can be found in Figure 1. Detailed instructions to use the tool are given in [8].

The input to the DNA-Storalator is a single design file in a text format which contains the input-strands to be simulated in the storage processes. The tool includes four components that are the crucial parts of any DNA-based storage system.

1) *Error characterization using SOLQC [8]*: There are several factors that affect the error characterization of a DNA storage library. The most important ones are the synthesis and the sequencing technologies. However, there are additional factors that affect the error rates, such as the design factors and the method that is used thru the PCR process. Therefore, when exploring both new and existing methods, it is required to characterize the errors of the library. Using a small sample

of a sequenced DNA storage library, the tool can characterize and analyze the errors in the library using the SOLQC tool [8]. Then, these error characteristics can be used to simulate additional data from the same synthesis and sequencing method.

2) *Synthesis, sequencing, and PCR error-simulation*: This component simulates the different errors (insertions, deletions, and substitutions) which occur in the chemical processes. For a combination of technologies of synthesis and sequencing methods, the user is presented with heuristics of the errors probabilities for the different types of errors and the different types of bases. The heuristic of these errors is based on results from previous experiments, such as [5], [9], [10]. The DNA-Storalator also allows user-defined error rates.

The PCR step is simulated by generating a different number of copies for every given designed strand. The number of copies in each cluster can be defined in the following two ways. a) Explicit definition - Using a vector that defines the exact number of copies in each cluster. b) Distribution - In this method, the user determines the probability density function of the cluster size distribution, the average, minimum, and maximum cluster size. Then, the tool simulates the clusters according to the defined distribution. The default distribution is the skewed-normal distribution.

To conclude, for each input-strand, the tool calculates the amount of copies to be produced according to the defined distribution. Then the simulator linearly scans the bases of every strand and inserts different errors according to the base type and the error rates. Lastly, the generated reads are shuffled.

3) *Clustering*: Following the error simulation, the user is able to perform the clustering step. The goal of this step, is to partition the unordered set of noisy copies into clusters, such that all the copies in each cluster originate from the same original designed input-strand. Currently, the DNA-Storalator includes three clustering algorithms. a) *Pseudo clustering algorithm*. In this algorithm the perfect clustering is given as an input. Then, the algorithm filters out from each cluster every read that has more errors than a given threshold. b) *Index-based clustering*. In this algorithm, the clusters are created by the reads' indices (with exact match or with some user-defined similarity threshold). Then, the similarity between the data part of the reads in each cluster is being evaluated with edit distance to filter-out reads. c) *Min-hash based algorithm [6]*. The DNA-Storalator also includes an implementation of the algorithm published in [6]. In this algorithm, several random hash functions are defined on the reads. Then, the hash functions are used to cluster together reads with similar hash values.

Following the clustering, several statistics are presented to the users. These statistics include the number of clusters generated, true-positive rates, and false-negative rates of the clustering stage. It is important to note that the tool conveniently allows users to perform integration with new clustering algorithms. By doing so, the user is able to test and compare different clustering algorithms.

4) *Reconstruction*: In this part it is assumed that the clustering step was completed, and the focus is on estimating the original strand from its noisy cluster. The input to this step is the clustered file of the noisy reads. Each cluster consists of noisy copies of a designed strand which are used for its recovery. The problem of using a set of erroneous sequences in order to recover the correct one falls under the framework of Levenshtein's *reconstruction problem* [4] and the *trace reconstruction problem* [1]. These models assume that the sequence is transmitted over multiple channels, and the decoder, which observes all channel estimations, uses this inherited redundancy to correct the errors. The main problem studied under this paradigm asks for the minimum number of channels that guarantees successful decoding either in the worst case or in high probability. The simulator includes several state-of-the-art reconstruction algorithms that can be performed either on clustered simulated data, or on any given clustered data sets. The simulator includes linear-time reconstruction algorithm [3] and dynamic-programing based reconstruction algorithms [7]. Additionally, the simulator also includes a trellis-based reconstruction algorithm [9], that can be performed if the error probabilities are known.

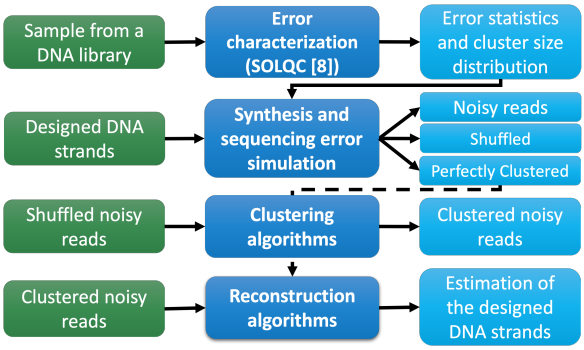

Figure 1. A schematic overview of the DNA-Storalator and its main components. This figure summarizes Section II.

III. Use-Case Examples

The list below describes some of the use-cases of the tool.

1) *Development of new coding techniques for DNA storage.* The tool allows users to analyze how different coding techniques affect the accuracy of the clustering and the reconstruction algorithms. Additionally, the simulator can be used to estimate the required error-correction capability of current/future DNA synthesis and sequencing methods.
2) *Development of new algorithms for DNA storage.* The tool supplies a convenient way to compare new and existing algorithms for DNA storage systems.
3) *Experiment designing with the DNA-Storalator.* The simulator provides an efficient method to test new algorithms and coding techniques before performing expensive and time-consuming wet experiments.

References

[1] T. Batu, S. Kannan, S. Khanna, and A. McGregor, "Reconstructing strings from random traces," *Proc. ACM-SIAM symp. on Discrete algorithms*, pp. 910–918, 2004.

[2] G. Chaykin, N. Furman, O. Sabary, D. Ben Shabat, and E. Yaakobi, "DNA-Storalator: End-to-end DNA storage simulator," https://github.com/gadihh/DNASimulator, 2022.

[3] P. S. Gopalan et al. "Trace reconstruction from noisy polynucleotide sequencer reads," *US Patent App.*, 15/536,115, 2018.

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

[5] L. Organick et al. "Random access in large-scale DNA data storage," *Nature Biotechnology*, vol. 36, pp. 242, 2018.

[6] C. Rashtchian et al. "Clustering billions of reads for DNA data storage," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[7] O. Sabary, A. Yucovich, G. Shapira, and E. Yaakobi, "Reconstruction algorithms for DNA-storage systems," *bioRxiv 2020.09.16.300186*, 2020.

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

[9] S. R. Srinivasavaradhan et al."Trellis BMA: Coded trace reconstruction on IDS channels for DNA storage," *Proc. Int. Symp. Inf. Theory*, pp. 2453–2458, 2021.

[10] S. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Scientific Reports*, vol. 7, no. 1, pp. 1-6, 2017.