Department of Biomedical Engineering

EINDHOVEN UNIVERSITY OF TECHNOLOGY

DBL Computational Biology

Storing digital data in DNA molecules

Research proposal

Project 7, Group 12

Dennis Brink	1442155
Justin Kleinveld	1430254
Allard Klerks	1320629
Tala Sabha	1327208
Fijten Vlasman	1268589

Tutor: Bas Bögels

Project coordinator: Bas Bögels

Eindhoven, November 26, 2021

1 Introduction

The amount of data in our world is steadily increasing. The world is collecting more data than it can store. Scientists are trying to create new methods that could increase the data capacity of a single DVD to one petabyte. One suitable method for storing data is DNA.[5] DNA is an attractive medium to store data in because it is extremely dense, with a theoretical limit above 1 EB/mm3, and a observed half-life of over 500 years. [1] The write process for DNA storage, maps digital data into DNA nucleotide sequences. Reading the data involves sequencing the DNA and decoding the information back to the original digital data.

Under ideal conditions, the capacity per nucleotide is 2 bits, as there are four possibilities. However, There are some restrictions. DNA sequences containing a lot of G, and C nucleotides or long homopolymers (extended sequences of repeated nucleotides) are difficult to synthesize correctly and have high chances of errors. This will be further discussed later. DNA synthesis and sequencing is far from perfect, with error rates on the order of 1% per nucleotide. Sequences can also degrade while stored, further compromising data integrity. Second, randomly accessing data in DNA-based storage is problematic, resulting in overall read latency that is much longer than write latency. The coupling efficiency of a synthesis process is the probability that a nucleotide binds to an existing partial strand at each step of the process. Although the coupling efficiency for each step can be higher than 99%, this small error can still results in an exponential decrease of product yield with increasing length and limits the size of oligonucleotides that can be efficiently synthesized to about 200 nucleotides.

Fountain codes are a class of codes with the purpose that an unlimited set of coding symbols can be generated from a given set of source symbols, so that the original source symbols can be recovered from any subset of the coding symbols that are of equal size or slightly larger than the number of source symbols. A fountain code is optimal if the used k-source symbols can be recovered from all k-coding symbols. [6]While writing the code it is very important that the function maps a key to the DNA pool where the strands that contain data reside.The research question is therefore : *How do you design an algorithm that uses fountain codes to encode binary data to DNA sequences and which can be decoded back to the original binary data after amplification and error correction?*

It is important to find a balanced trade-off between storage density, reliability, and performance. A DNA storage system consists of a DNA synthesizer that encodes the data to be stored in DNA, a storage container with compartments that store pools of DNA that map to a volume, and a DNA sequencer that reads DNA sequences and converts them back into digital data. The DNA strands will be stored in “pools” that have stochastic spatial organization and do not permit structured addressing. Therefore, it is necessary to embed the address itself into the data stored in a strand. A storage system needs a way to assign identification tags to data objects so they can be retrieved later.

2 Requirements

Biochemical processes like DNA synthesis are prone to errors. Errors such as PCR dropout, degradation of DNA over time and sequencing errors such as insertions or swapping of nucleotides could pose a major issue while decoding data from DNA. It has been shown that the error pattern in DNA largely depends on the input sequence. The algorithm should account for these biochemical limitations while encoding data to nucleotides in such a way that the chances of errors occurring during synthesis are minimized. Prior work has already shown that it is possible to recover data from DNA with a small amount of error with the use of proper encoding and decoding schemes.[7]

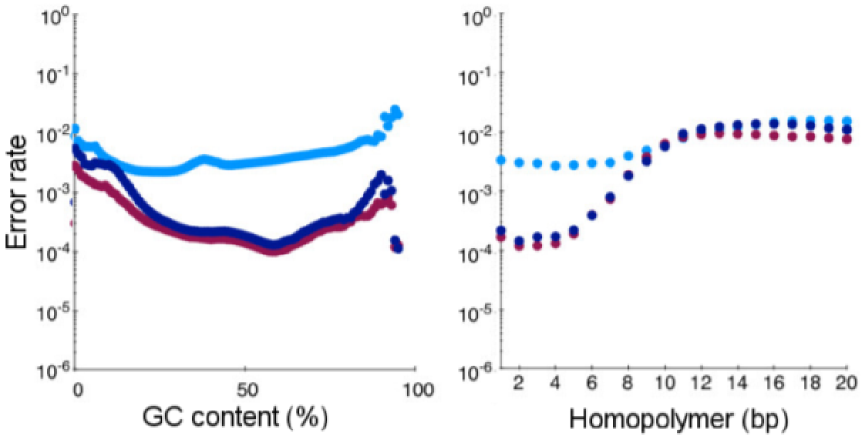

Figure 1: Error rates of Illumina sequencing as a function of GC content and homopolymer length. Light blue: mismatches; dark blue: deletions; purple: insertions. [9]

Figure 1 shows that the error rates for Illumina sequencing are lowest with a GC content of about 50%. Therefore a GC content of 45% to 55% is accepted within sequences after encoding as the error rate for that percentage is acceptable. [8, 9]
Furthermore, Figure 1 shows that sequences with longer homopolymer regions have increasing chances for errors. Error rates seem to be at a minimum for homopolymers of up to four nucleotide, so the encoding of the data should be in such a way that it would be impossible for sequences to have more than four of the same nucleotide in a row.[2, 9]

Modern synthesizing techniques have been able to produce sequences with a maximum of 200 nucleotides with an acceptable error rate. As it is desired to lower the error rate while also having an acceptable redundancy most prior researches had sequences of lengths between 150 and 160 nucleotides, which will also be adapted in this research.[3]
To make sure every sequence can be amplified to increase redundancy, it is mandatory that the primers used are not self-complementary. Self-complimentary would lead to primers sticking to themselves and hindering PCR amplification. This would result in the loss of information of that sequences which would be unfavorable. To prevent this intra-primer homology the maximum complimentary nucleotides should be three.[7]

Data preserved in DNA is not written to a specific location, as of such a seed should be added to each sequence to be able to link it to its original position in the binary datafile. [3, 8]

All the points mentioned above will be taken into consideration while designing the algorithm for data storage in DNA.

3 Proposed methods

The following algorithm is proposed to meet these requirements.
Fountain codes, specifically in the form of Luby-transform codes, are used to encode the binary data into so called droplets. The Luby-transform works by segmenting the data into segments of equel length. A XOR operator is then used on these segments randomly and a seed is added which can be used to determine on which segments a XOR operator was used to create the droplet. This process can be seen described in Figure 2. This combination of modified data and seed forms a droplet. This method enables the creation of a practically infinite amount of different droplets and, if properly tuned, as long as the total data contained in the collected droplets is at least 3% more than in the original data [4], they can be decoded back into the full data.
The Luby-transform is chosen because its relatively low complexity allows for straightforward implementation for binary to DNA encoding. The droplets are encoded into DNA using a naive encoding as shown in table 1.

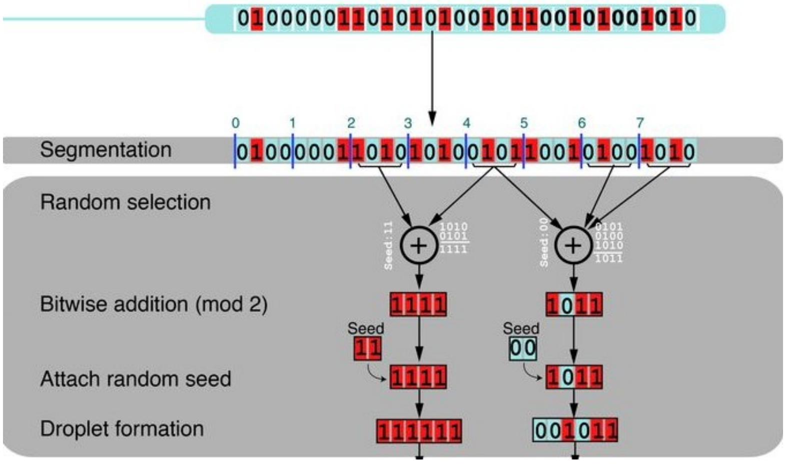

Figure 2: Formation of droplets using the Luby-transform [4]

Table 1: 2-bit to DNA encoding scheme

Bit	Base
00	A
01	C
10	G
11	T

In order to ensure that this does not result in DNA segments that are not feasible for synthesis or sequencing, due to not following the before stated requirements, checks are performed and any non-feasible segments are discarded. This method of encoding data enables many erasures to be corrected easily, preventing errors in synthesis from causing problems in reading the data.

Separate errors that occur during PCR are much more complicated to correct using fountain codes. Therefore, further error correction in the form of Reed-Solomon codes is implemented as well. First the data will be encoded into droplets, and then Reed-Solomon codes will be added to each droplet. When decoding this will enable the algorithm to fix errors in the droplets, and then decode the droplets back into the original data. Reed-Solomon codes are expected to perform well for this task, because, they are part of the Maximum Distance Separable (MDS) codes. This means that they provide the maximal possible error detection/correction. [10] Other error-correcting codes with this property do exist, but provide little benefit outside of additional computational speed, which is irrelevant for the purposes of DNA storage currently. Therefore, the widespread use of Reed-Solomon codes makes them preferable. In order to ensure that the amount of information per nucleotide can remain high, it is preferable to use as little as possible error correcting symbols with Reed-Solomon, therefore a low error sequencing method such as Illumina sequencing is preferred.

A number of pre-processing steps are also necessary to enable this process to be performed smoothly on a wide variety of data. Firstly, padding will be added to make sure the file can be evenly segmented into segments for the creation of droplets using the Luby-transform. This padding will be removed during decoding and in order to facilitate this a number of bits are added to the start of the data to indicate how much padding was used. Secondly, the data will be encrypted using a stream cipher with a set key in order to randomize the data. This is helpful because data consisting almost entirely of one digit would cause the variety of created droplets to drop significantly. Since the droplet corresponding to only ones or only zeros is not feasible for storage, this would case the majority of created droplets in this case to be non-feasible. In the case where the data consists entirely of one digit, this would even make it impossible to encode. Therefore the data is randomized before applying the Luby-transform. The full process of encoding and decoding can be seen summarized in figure 3 shown below

Based on previous research[4], we hypothesize that the algorithm as described above could approach, but not reach, the theoretical limit for Shannon information per nucleotide of 1.98 bits/NT. While still providing robustness against data corruption due to the inclusion of both fountain codes and Reed-Solomon codes.

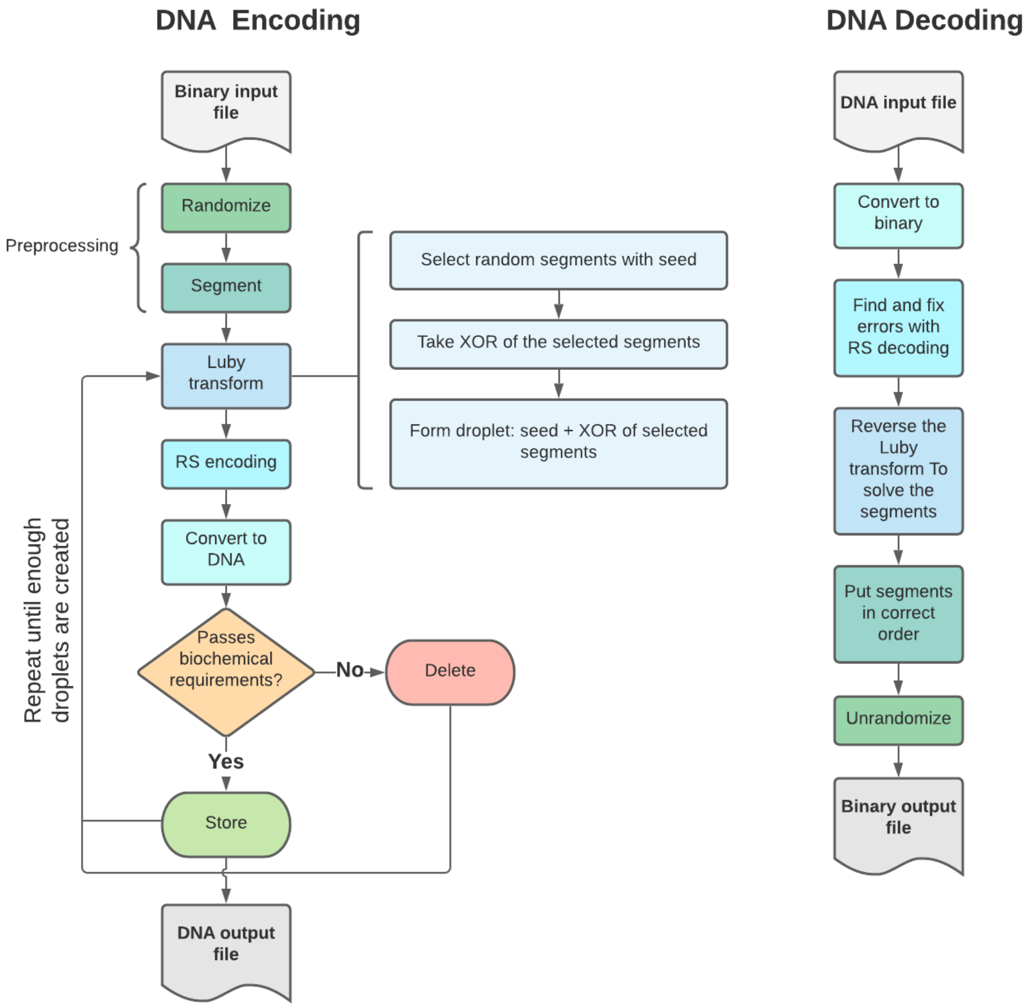
DNA Encoding
Binary input file
Preprocessing
Randomize
Segment
Luby transform
Select random segments with seed
Take XOR of the selected segments
Form droplet: seed + XOR of selected segments
RS encoding
Convert to DNA
Repeat until enough droplets are created
Passes biochemical requirements?
No
Delete
Yes
Store
DNA output file
DNA Decoding
DNA input file
Convert to binary
Find and fix errors with RS decoding
Reverse the Luby transform To solve the segments
Put segments in correct order
Unrandomize
Binary output file

Figure 3: Steps necessary for encoding and decoding data in DNA

4 Planning

In order to finish the work in time a planning has been created. This planning contains all necessary building blocks for a working algorithm. There are several risks to the planning. The most obvious one is failing to get a part of the algorithm working. Another risk would be having a wrong estimate for the amount of work certain parts require. One of the most dangerous risks comes from a fault in the algorithm's architecture. If the requires a function to work which wasn't planned on a whole new task would suddenly have to be completed. In order to mitigate these risk the last week has been kept empty. This way there is room for tasks running long or new tasks having to be completed. The goal is to make the algorithm as broad as possible and to create DNA sequences that require little redundancy. Should it be clear that the planning cannot be achieved sacrifices in redundancy and broadness can be made.

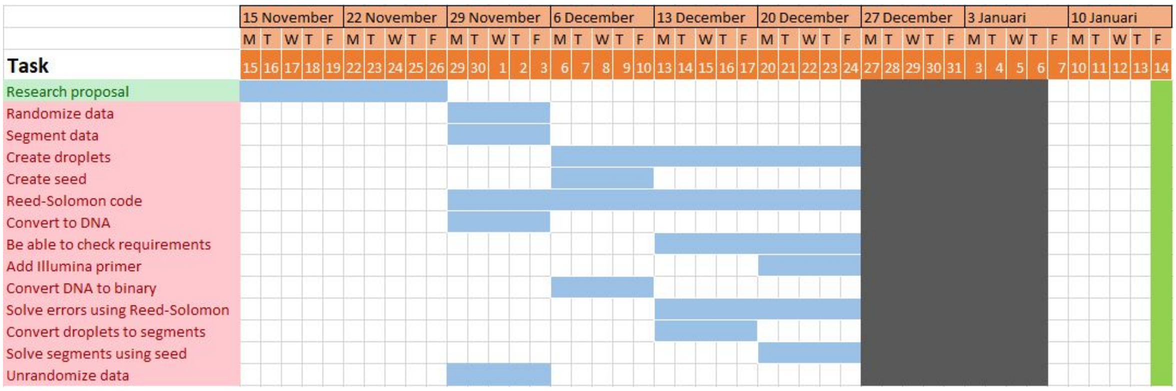

Figure 4: Gantt chart of the planning

References

[1] Matthew Harker David Haile James Oskam Charlotte L Hale Marie L Campos Paula F Samaniego Jose A Gilbert M Thomas P Willerslev Eske et al Allentoft, Morten E Collins. The half-life of dna in bone: measuring decay kinetics in 158 dated fossils. *Proceedings of the Royal Society B: Biological Sciences*, 279(1748):4724–4733, 2012.

[2] Guruprasad Ananda, Erin Walsh, Kimberly D Jacob, Maria Krasilnikova, Kristin A Eckert, and Kateryna D Chiaromonte, Francesca Makova. Distinct mutational behaviors differentiate short tandem repeats from microsatellites in the human genome. *Genome biology and evolution*, 5(3):606–620, 2013.

[3] Sheridan Ashlock, Daniel Houghten. Forward error correction for dna data storage. *Procedia Computer Science*, 80:1011–1022, 2016.

[4] Dina Erlich, Yaniv Zielinski. Dna fountain enables a robust and efficient storage architecture. *Science*, 355(6328):950–954, 2017.

[5] Miłosz Grucha A. Komputery biomolekularne oparte na bramkach logicznych za pomoca dna. wpływ komputerów biomolekularnych na aspekty ekonomiczne. *Studia Informatica Pomerania*, 2017.

[6] Khaled F Hayajneh and Mehrdad Yousefi, Shahram Valipour. Improved finite-length luby-transform codes in the binary erasure channel. *IET Communications*, 9(8):1122–1130, 2015.

[7] G.M. Kosuri, S. Church. Large-scale de novo dna synthesis: technologies and applications. *Nature Methods*, 11:499–507, 2014.

[8] L. Organick and YJ. Ang, S. Chen et al. Random access in large-scale dna data storage. *Nature Biotechnology*, 36:242–248, 2018.

[9] Michael G Ross, Carsten Russ, Maura Costello, Andrew Hollinger, Niall J Lennon, Ryan Hegarty, and David B Nusbaum, Chad Jaffe. Characterizing and measuring bias in sequence data. *Genome biology*, 14(5):1–20, 2013.

[10] R. Lekh. Vermani. *Elements of Algebraic Coding Theory*. Chapman Hall, 1996.