IET Nanobiotechnology

Research Article

IET Journals
The Institution of Engineering and Technology

Encoding scheme for data storage and retrieval on DNA computers

ISSN 1751-8741
Received on 24th April 2020
Revised 22nd June 2020
Accepted on 10th July 2020
E-First on 2nd September 2020
doi: 10.1049/iet-nbt.2020.0157
www.ietdl.org

Dolly Sharma[1] ✉, Ranjit Kumar[2], Mayuri Gupta[1], Tanisha Saxena[1]

[1]*Department of Computer Science and Engineering, School of Engineering, Shiv Nadar University NCR, India*
[2]*Amity Institute of Nanotechnology, Noida, Amity University Uttar Pradesh, India*
✉ *E-mail: dolly.sharma@snu.edu.in*

Abstract: There has been exponential growth in the amount of data being generated on a daily basis. Such a huge amount of data creates a need for efficient data storage techniques. Due to the limitations of existing storage media, new storage solutions have always been of interest. There have been recent developments in order to efficiently use synthetic deoxyribonucleic acid (DNA) for information storage. DNA storage has attracted researchers because of its extremely high data storage density, about 1 exabyte/mm^3 and long life under easily achievable conditions. This work presents an encoding scheme for DNA-based data storage system with controllable redundancy and reliability, the authors have also talked about the feasibility of the proposed method. The authors have also analysed the proposed algorithm for time and space complexity. The proposed encoding scheme tries to minimise the bases per letter ratio while controlling the redundancy. They have experimented with three different types of data with a value of redundancy as 0.75. In the randomised simulation setup, it was observed that the proposed algorithm was able to correctly retrieve the stored data in our experiments about 94% of the time. In the situation, where redundancy was increased to 1, the authors were able to retrieve all the information correctly in the proposed experiments.

1 Introduction

Adenine, thymine, cytosine and guanine are four different nucleotides that form the building blocks of deoxyribonucleic acid (DNA) arranged in a double helix structure. Adleman [1] demonstrated an in vitro solution to the directed Hamiltonian path problem. Adleman's experiment attracted many researchers into further exploring DNA computing because of its huge potential in solving mathematical and computational problems and also due to its capacity to hold data, ability to perform computation with extremely high parallelisability. DNA Computers are Biochemical Nano computers that solve computational problems in a test tube. This alternate computing model uses molecular biology and has many advantages over silicon-based computing. Several studies have been presented in the past that explore the different facets of DNA Computing [2–4].

DNA computing can be classified into three categories [5]: (i) intramolecular: they are usually involved in the later stage of exploring candidates and arriving at the final solution. An example is successive localised polymerisation designed by Sakamoto *et al.* [6], (ii) intermolecular and (iii) supramolecular: such as molecules binding to form self-assembled structures and also influence the protein binding. Chen and Ellington [7] discussed the pros and cons of DNA computing. The direct role of DNA computing in the biochemical systems seems very promising in the future such as applications to smart drugs, drug design, drug release as well as DNA self-assembly. They also present the challenges associated with this model such as dealing with error, kinetics, etc. Winfree *et al.* [8] presented that using hybridisation it is possible to simulate a computation as a one-dimensional cellular automata and also shown disjunctive hybrid probability logic programs (DHPP) can be solved in a test tube using multiple tiling reactions. They call these reactions as one-pot reactions.

This work is focused on designing efficient data encoding scheme for DNA computing. Although numerous articles have been published in the past decades in the field of DNA computing, there still exist numerous challenges that need to be addressed at the implementation level. One such challenge would be the cost and technology required to commercialise this alternate form of computation. Interpretation of the results and visualisation are also hurdles that need to be efficiently crossed in order for the technology to be accepted and adopted. Errors in biochemical computations are another big challenge, which may occur due to undesired annealing [4].

DNA computing has the capability of performing basic arithmetic and Boolean operations. Frank Guarnieri *et al.* [9] implemented the addition operation on single bits for the first time using biomolecular computing in recombinant DNA. Non-negative binary numbers were represented using primer extensions to perform addition but were limited to two numbers only and this method was also limited to only single computation as it neither exploited parallelism nor it had the capability to perform repetitive computation. Some researchers also presented basic arithmetic operations and allowed chaining [10, 11]. Hasudungan *et al.* [12] presented an algorithm to solve the minimum vertex cover problem using DNA computing. The first step in this algorithm was to design an encoding technique and the second step was to design a bio-operation technique. They demonstrated the working of their algorithm using a sample graph with six vertices and six edges.

DNA computing-based solutions have been proposed for complex computational problems, such as Hamilton path problem [1], maximal clique problem [13], satisfiability problem [14] in only 91 steps. The solutions exploit massive parallelism to achieve good scalability as well as speedup. The satisfiability problem [14] is solved by encoding all candidate solutions using DNA molecules and attached to the surface. The candidates that do not form a solution are identified with the help of several cycles of hybridisation operations and exonuclease digestion. In the end, polymerase chain reaction (PCR) is used to amplify the remaining molecules containing the solution. Lipton [15] proposed a solution for two-variable SAT problem and also claimed that all problems in the NP class can be reduced to the satisfiability problem. Sakamoto *et al.* [6] presented a solution to NP-complete problems using a parallel overlap assemble of the single layer of successive transitions. A single-stranded DNA molecule's 3′-end sequence encodes for the current state of the machine. The current state is annealed to the desired area of DNA such that the desired next state is achieved and encoded on the 3′ end. In this work, the authors propose to solve NP-complete problems using the method. They present two experiments first using isothermal reactions and

second using unnatural bases to avoid out-of-frame annealing that can be used in DNA computing.

Security has to be on the highest priority when dealing with such voluminous data. Various encryption techniques based on DNA Computing have also been proposed in the past. Acharya [16] proposed an algorithm using the concept of permutation-diffusion for faster image encryption. They implemented the diffusion process using the AES algorithm and show that the algorithm is fast as well as secured. Mondal and Mandal [17] presented a scheme for encryption of greyscale images and proposed that the technique could also be extended to colour images. Hermassi *et al.* [18] proposed an encryption algorithm that combines a DNA addition with a chaotic map for encryption of a greyscale image. They claim that the result is non-invertible. However, illegal decryption of the ciphered image can be done with temporary access to the machinery.

In this work, we present an encoding and decoding scheme that is simple to implement, it is highly scalable and allows the user to choose the tradeoff between storage density and reliability. We repeated the simulations for each data 10 times using 0.75 redundancy and we could observe that in 94% of our experiments, we could retrieve the original string correctly. The minimum error observed during experiments over 10 experiments is 0%.

2 Background

Information stockpiling, called data storage, is a general term for chronicling information in electromagnetic or different structures for use by a PC or gadget. DNA and RNA, penmanship, phonographic recording, attractive tape and optical circles are the instances of capacity media. Data storage in a computerised, machine-coherent medium is now and again called digital data. New innovations and tech hypothesis advance the persistent extension of data storage capacity. New strong state drives can hold colossal measures of information in an exceptionally little gadget, empowering different sorts of new applications for some ventures, just as shopper employments. Cloud administrations and other new types of remote stockpiling likewise add to the limit of gadgets and their capacity to get to more information without structure extra data storage into a gadget [19]. Notwithstanding this improvement, putting away zettabytes of data would even now take a large number of units, and utilise critical physical space. However storage density is only one form of archival. Current long haul recorded capacity arrangements require revives to clean defiled information, to supplant broken units, and to invigorate innovation. If we have some ease of preserving the world's data, we have to look for important advances in storage density and durability. Researchers have been putting away digital information in DNA since 2012. That was when Harvard University geneticists encoded a 52,000-word book in a large number of scraps of DNA, utilising strands of DNA's four-letter letters in order of A, G, T and C to encode the 1 s of the digitised document [20]. Their specific encoding plan was generally wasteful, notwithstanding and could store just 1.28 petabytes/g of DNA. Different methodologies have improved. Be that as it may, none has had the capacity to store the greater part of what analysts figure DNA can really deal with, about 1.8 bits of information per nucleotide of DNA. The volume of information that can be combined today is constrained for the most part by the expense of combination and sequencing, however, development in the biotechnology industry forecasts requests of extent cost decreases and proficiency upgrades. Banal *et al.* [21] recently presented a strategy that could provide random access to large archival files. Since they encapsulated the data in silica particles, the density of data storage is higher than PCR-based methods.

A DNA storage framework must conquer a few difficulties. First, the cost associated with synthesis and sequencing. The expenses per megabyte were evaluated at $12,400 to encode information and $220 for recovery [22]. In any case, it was noticed that the exponential abatement in DNA amalgamation and sequencing costs, on the off chance that it proceeds into the future, should make the innovation financially savvy for long haul information stockpiling inside around ten years. Second, error rates on the order of 1% per nucleotide. Sequences can likewise corrupt while putting away, further trading off information integrity. A key part of DNA storage is to devise fitting encoding plans that can endure errors by including repetition [23]. Existing methodologies have concentrated on repetition yet have overlooked thickness suggestions. Third, random access to DNA-based data storage is dangerous, that can result in the read latency that is actually much longer than write latency. Also, the current DNA-based data storage system does not have the capability of synthesising long strands. For DNA-based data storage to be a feasible choice of data storage, the automation of the entire process of reading and writing should be done. The recent methods and technologies are considered to be time consuming and error prone. Luckily, there's a steady pace of development discovering more up to date and better answers for these relentless data storage issues.

2.1 Review of encoding schemes

Several data encoding schemes have been proposed in the past [24–26]. Along with efficient storage, these encoding schemes also need to hide data in the DNA to ensure security. Bornholt *et al.* [25] presented a DNA-based archival storage system, where they implemented Goldman encoding and XOR encoding on four image files and observed that random access can be effectively implemented on DNA storage. The authors propose that a new payload is created using XOR operation over two existing payloads. The address of this strand encodes the address of original strands as well as information whether the strand is the original one or an XOR. They were able to recover three of the files without any problem but one of the files had an error and needed to be corrected to be recovered. The packing density is higher with 1.5 repeats of each nucleotide, which is comparatively lower than many others. The algorithm proposed in Bancroft *et al.* [26] could successfully recover a message of length 106 using simple ternary encoding; where messages were composed of 26 English alphabets and space. However, this method is not scalable. Goldman *et al.* [27] presented an encoding scheme with four-fold redundancy by overlapping segments, where each window represents an output strand. The encoding scheme is not very compact due to the fact that it uses tunable redundancy, but it successfully stores and retrieves the data. Shimanovsky *et al.* [28] propose a method to hide data in non-transcribed DNA and non-coding RNA. Takahashi *et al.* [29] proposed an automated DNA storage mechanism where they demonstrated automated writing, coding and reading of 5-byte data. Organick *et al.* [30] proposed PCR-based method that could provide random access to data. They showed that retrieval is possible with an average copy number of 10 and a packing density of 17 exabytes/g which provided significant improvement.

2.2 Machine learning using DNA computing

Neural network computation is based on a simplified model of how the human brain works; containing billions of neurons each connected to thousands of synapses. A neuron fires and sends a signal to other neurons when it reaches a threshold potential, where the potential is computed on the signals received from other neurons. Artificial neural networks are an attempt to mimic how the human brain learns. Qian *et al.*[31] and Cherry and Qian [32] presented a method to perform neural network computation, more specifically the linear threshold gate, using strand displacement cascades. Seesaw gates are used to build these linear threshold gates which are further used to construct multilayer neural networks. The seesaw gates have one or more connections on both of the sides. In this model, a Reporter gate is used to collect the output signal. Hopfield neural networks [33] have neurons that are fully connected to each other and in best-case scenarios they are able to remember a set of patterns. The network learns from these set patterns as if they are stored in memory. When it receives a piece of information such as a partial or distorted pattern, it identifies and returns back the correct pattern from its memory. The authors have shown that computational neural networks can be implemented in vitro using transcriptional circuits. The building blocks for these transcriptional circuits are transcriptional switches that transcribe efficiently only in the presence of an activation.

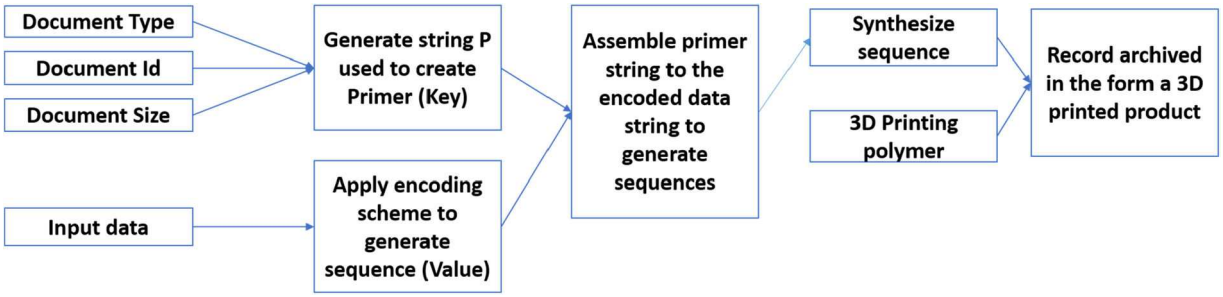


Fig. 1 *Block diagram to present the complete data storage mechanism*

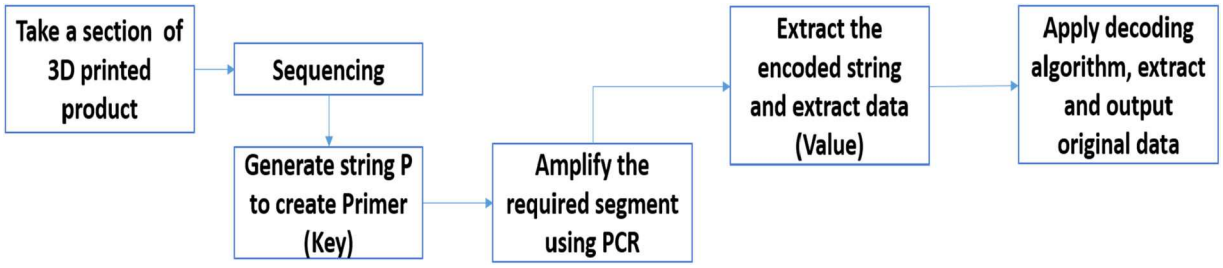


Fig. 2 *Block diagram to present the complete data retrieval mechanism*

Here the input is the activator, the output is the transcription concentration. However, promising these theoretical models may sound, but it is essential to build a fault-tolerant system as strands may have undesired reactions which might result in incomplete or even incorrect products. It was observed that by restoring the output values in digital form resulted in less noise. However, it is still an open challenge to design fully fault-tolerant biochemical neural networks [9–11].

Evolution in computing is termed 'evolutionary computation' and evolution biomolecules is called '*in vitro* evolution'[34]. Evolution is a concept we tried to understand from nature and then model similar approaches for solving computational problems. Evolutionary computation has a lot of application from solving complex computational problems, understanding behaviour of involved entities etc. The advantage of using DNA computing for evolutionary computation has many benefits, such as: (i) process in parallel and generate higher-fitness for a larger population, (ii) huge capacity to store information and (iii) capability to perform crossover. The authors have addressed the MAX 1 s problem in this work. Ren *et al.* [35] used VDNA-encoding method and genetic algorithm to design a new algorithm named VDNA-EA. It is capable of implementing the T-S fuzzy controller and optimisation of the parameters. The authors have simulated the experiment. Yoshikawa *et al.* [36] present the 'DNA encoding method' using pseudo-bacterial genetic algorithm, which is based on the concept of recombination in bacteria. Deaton *et al.* [37] present an evolutionary algorithm. Such algorithms are more suitable for DNA computing as errors do not cause an incorrect result, rather they code for variation in the population. They implement this Evolutionary algorithm to search for good DNA encodings. Li *et al.* [38] proposed a genetic algorithm approach to solve the maximal clique problem. They tested the randomly generated problems and showed that DNA computing produced correct answers within a few iterations.

Direct proportional length based DNA computing approaches for shortest path problem was proposed by authors in [39–41]. The process initialises with the generation of five unique single-stranded DNA sequences that represent nodes in a graph. In the next step, length constraints are formed and then the remaining sequences are generated. The various approaches differ in the method used to generate the DNA sequences such as graph method, Generate and test method and population-based ant colony optimisation method.

Games theory has a lot of very important applications and the capability of DNA computing in playing games would be extremely helpful in the game-theoretic approaches. Wood [42] proposed a method where DNA can play poker. They use a simplified version of poker incorporating bluffing, calling and folding. DNA is capable of encoding for the different players where the player and the dealer independently develop their strategies. The strategies are learnt using multiple instances of the game, similar to the Genetic algorithm where selection is based on the fitness function. Cukras *et al.* [43] present an RNA-based model for playing chess. The authors demonstrated a solution for Knights problem on a 3X3 chessboard. The results were correct most of the time as compared to the random chance. However, the authors did emphasise on the necessity of handling the error in DNA computing in order to get correct solutions almost all the time. Macdonald *et al.* [44] demonstrated Tic-Tac-Toe game using DNA computing. Wood [42] presented a study on Game theoretic approaches for biomolecular computing.

3 Encoding scheme for data storage and retrieval

In this work, we present a data encoding and retrieval scheme. From the review in the previous sections, it is evident that this technology is extremely efficient for storing archival data. Fig. 1 represents the complete process for data encoding and storage and Fig. 2 presents the complete process for decoding and retrieval. The encoding process, uses two separate inputs: (i) encoded data (value) and (ii) primer (key). In order to retrieve the required data correctly, the key is created using the index of the data in the database, the type of data and size of the file in the number of bits. This key is then encoded using the proposed scheme to create a key. The significance of inserting the key is to make indexing possible in the synthetic DNA as well. Further, the key and the value are stitched together and sequenced and stored in the form of a 3D printed material [45]. The process is reversible as the targeted data can be identified using the primer and decoded by the retrieval algorithm (Table 1).

3.1 DNA encoding scheme

The data encoding scheme presented in this section can be used to encode all kinds of digital data such as text, images, graphs, etc. The fragmentation process used here is presented in Fig. 3 and the XOR operation applied to the process can be observed in Figs. 4 and 5.

3.2 DNA retrieval scheme

The retrieval algorithm presented in this section decodes the synthetic DNA and outputs the original string. In the case where we create 0.75-time redundancy, we retrieve the original string in

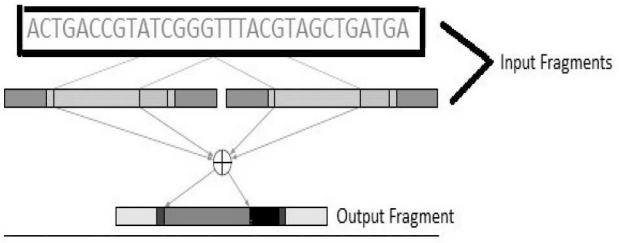


Fig. 3 *XOR operation*

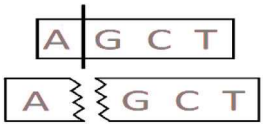


Fig. 4 *DNA fragmentation*

Encoding Algorithm
Input: Data to be stored in DNA (Str)
Output: Encoded Data
SynDNA = Null
Convert Str to equivalent binary code BinStr
N = lengthOfBinStr
For (I := 1, I ≤ N; I+=2)
{ Convert binary pair at I to corresponding
Nucleotide Nuc
DNA := DNA + Nuc
}
//Create Fragments
Fragment1 := Null
Fragment2 := Null
Fragment3 := Null
For(J=1 to lengthofDNA)
{ Fragment2 := Fragment1+DNA[J]
Fragment1 := Fragment2 + DNA[J+1..J+3]
}
//Encoding
I = 0
Repeat
{ for (c = 1, c<=3, c++)
{ Oper = Fragment1[I+c] ⊕ Fragment2[I]
Fragment3 = Fragment3+Oper
}
I = I+c
c=0
} Until (I ≤ |Fragment1|)
Prepare final encoding of Fragment1, Fragment2 and Fragment3 and store

Fig. 5 *Encoding algorithm*

three ways and resolve any observed errors. To improve efficiency, the redundancy can be increased (Fig. 6).

3.3 Application of the proposed approach

One interesting application of this data encoding and decoding scheme can be in the area of health care. Health care industry stores an increasingly high amount of structured as well as unstructured data [46]. Electronic health records such as patient details, billing information, laboratory reports, hospitalisation records, medication records are more structured, whereas doctor's handwritten notes are an example of unstructured data. Medical imaging data such as computed tomographic scans, magnetic resonance imaging, X-rays, ultrasounds etc. also require a huge amount of storage space. Big data analytics in health care will assist in aggregating large amount of medical data collected over the years into medical databases while payers and providers have digitised their patient's records. Traditionally, patient's medical records are stored in physical paper files, X-ray films etc. We need a system that is able to store the health care data in large volumes using an efficient, robust as well as a secured mechanism [47, 48].

Table 1 XOR operation results

Inputs	Output
A⊕A	A
A⊕C	C
A⊕G	G
A⊕T	T
C⊕C	A
C⊕G	T
C⊕T	G
G⊕G	A
G⊕T	C
T⊕T	A

Retrieval Algorithm
Input: Encoded Data (Fragment1, Fragement2, Fragment3)
Output: Original Data (Str)

//Decoding
Temp1=Null
Temp2=Null
I = 1
Temp1 = Null
Repeat
{ oper = Fragment1[I] ⊕ Fragment3[I]
Temp2 = Temp2 + oper
} Until (I ≤ |Fragment1|)

I=1
Repeat
{ for (c = 1, c<=3, c++)
{ oper = Fragment1[I+c] ⊕ Fragment2[I]
Temp = Temp + oper
}
I=I+c
C=0
} Until (I ≤ |Fragment1|)

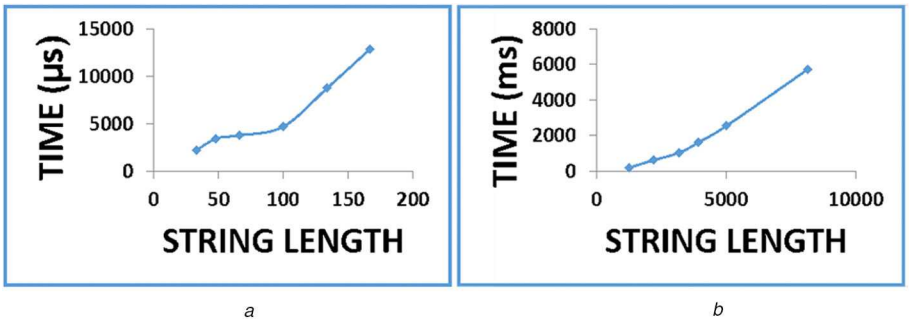
Compute DNA1, DNA2 and DNA3 using Fragment1, Fragment2 and Fragment3, Temp1 and Temp2

If (DNA1 = DNA2 = DNA3)
Output SynDNA
Else resolve conflicts and resolve SynDNA

Convert SynDNA to equivalent binary code BinStr

Convert BinStr to Str

Fig. 6 *Retrieval algorithm*



a b

Fig. 7 *Performance evaluation of the algorithm*
(a) Performance of random data sets, ***(b)*** Performance on news articles

Table 2 Performance on random data sets

String size	DNA length	Fragments 1&3	Fragment 2	Min error, %
33	132	99	33	0
48	192	144	48	0
66	264	198	66	0
100	400	300	100	0
134	536	402	134	0
167	668	501	167	0

Table 3 Performance on news articles

String size	DNA length	Fragments 1&3	Fragment 2	Min error, %
3171	12,684	9513	3171	0
1248	4992	3744	1248	0
3918	15,672	11,754	3918	0
2206	8824	6618	2206	0
8161	32,644	24,483	8161	0
4992	19,968	14,976	4992	0

Table 4 Performance on images

String image Id	DNA length	Fragments 1&3	Fragment 2	Min error, %
1	40,012	30,009	10,003	0
2	17,840	13,380	4460	0
3	17,052	12,789	4253	0

4 Experiments and results

For the encoding and decoding process to be acceptable by various domains, it is very important that it is capable of storing all types of data such as text, graphs, images, etc. In this work, we experimented with text, graphical representation of data and images. The algorithm was implemented in Java and the performance of these algorithms was tested using computer simulations. Three existing images and randomly generated strings and graphs of different sizes were generated to be encoded, stored and successfully retrieved. The randomly generated strings were of lengths 33, 66, 48, 100, 134 and 167. An example data set of each type is given in the Appendix separately. The experiments were implemented on Intel i5-7200U processor with 8 GB RAM and 2.71 GHz frequency and the performance with respect to time is shown in Fig. 7. The encoded sequence of bases generated by the first algorithm was passed as a parameter to our data retrieval algorithm. Since it has been shown in the literature [49] due to the biochemical constraints observed in the sequencing process, the Shannon capacity of a nucleotide is 1.98 bits, which comes to about 1% error. Thus, to test whether this scheme would be able to retrieve data with error, we randomly introduced 1% error in the fragments before passing the sequence to data retrieval algorithm.

The minimum error rates in the retrieved results can be observed in the results presented at Tables 2–4, when we randomly introduced an error of about 1% in the strands. In our experiments, for a string of length n, we require $n + 0.75n$ amount of storage. Also, the encoding and retrieval scheme is linear in nature. To generate the final strand, we notice the degeneracy for all the DNA sequence that could be obtained by pairing up any two of the three fragments that we obtained during encoding. We notice that in none of our experiments with graphs, do all of the three supposed DNA generated differ at a particular index. This fact has been used to predict the actual DNA. However, in text and images, it was observed that during some of the simulations, there was a small error. In repeated experiments, we observed that about 6% of our experiments showed that there was an error in the predicted string. So, in situations where reliability cannot be compromised at all, the error can be further reduced by creating more redundancy. In storing a string of size n, when we increased the redundancy to $1*n$ and further to $1.75*n$ in our experiments, we were able to correctly retrieve all the data stored. In addition, more fragments can be stored for that particular data by performing an XOR with the fragment containing second third and fourth base of the codons as well. Thus, this scheme gives the user an option to choose a trade-off between storage density and reliability.

5 Conclusion and future work

We have proposed algorithms for data encoding and data retrieval in this paper. We can store a large amount of data in DNA with almost no error if synthesising and sequencing are done efficiently. Different endeavours are in progress to investigate the capability of DNA to store cryptographic keys and other private data. The expense and designing obstructions to a feasible capacity of non-hereditary information in DNA are considerable and this innovation is in its outset. Conquering these hindrances would realise a transformation in data storage and security, enabling huge measures of information to be put away safely in only small amount of DNA. Researchers are developing new techniques to make DNA a secure storage medium.

6 References

[1] Adleman, L.M.: 'Molecular computation of solutions to combinatorial problems', *Science*, 1994, **266**, (5187), pp. 1021–1024
[2] Kari, L.: 'DNA computing: arrival of biological mathematics', *Math. Intell.*, 1997, **19**, (2), pp. 9–22
[3] Kari, L., Seki, S., Sosík, P.: 'DNA computing-foundations and implications', in Rozenberg, G. (Ed.): '*Handbook of natural computing*' (Springer, Berlin Heidelberg, 2012), pp. 1073–1127
[4] Pisanti, N.: 'DNA computing: A survey', *Bulletin of the EATCS*, 1998, **64**, pp. 188–216
[5] Ezziane, Z.: 'DNA computing: applications and challenges', *Nanotechnology*, 2006, **17**, (2), p. R27
[6] Sakamoto, K., Kiga, D., Komiya, K., *et al.*: 'State transitions by molecules', *BioSystems*, 1999, **52**, (1–3), pp. 81–91
[7] Chen, X., Ellington, A.D.: 'Shaping up nucleic acid computation', *Curr. Opin. Biotechnol.*, 2010, **21**, (4), pp. 392–400
[8] Winfree, E., Yang, X., Seeman, N.: 'Universal computation via self-assembly of DNA: some theory and experiments', in Landweber, L.F. (Ed.): '*DNA based computers II*' (American Mathematical Society, USA., 1996), pp. 191–213
[9] Frank Guarnieri, C.B.: 'Use of a horizontal chain reaction for DNA-based addition', in Landweber, L.F. (Ed.): '*DNA based computers II*' (American Mathematical Society, USA., 1996), pp. 105–111

[10] Leete, T., Klein, J., Rubin, H.: 'Bit operations using a DNA template'. Proc. of the Third DIMACS Workshop on DNA-based Computers, Philadelphia, USA., June 1997), pp. 159–166
[11] Klein, J.P., Leete, T.H., Rubin, H.: 'A biomolecular implementation of logically reversible computation with minimal energy dissipation', *Biosystems*, 1999, **52**, (1–3), pp. 15–23
[12] Hasudungan, R., Pangestuty, D.M., Latifah, A.J.: 'Solving Minimum Vertex cover problem using DNA computing', Journal of Physics: Conf. Series, Medan, Indonesia, Vol. **1361**, November 2018, pp. 012038
[13] Ouyang, Q., Kaplan, P.D., Liu, S., *et al.*: 'DNA solution of the maximal clique problem', *Science*, 1997, **278**, (5337), pp. 446–449
[14] Liu, Q., Wang, L., Frutos, A.G., *et al.*: 'DNA computing on surfaces', *Nature*, 2000, **403**, (6766), pp. 175–179
[15] Lipton, R.J.: 'DNA solution of hard computational problems', *Science*, 1995, **268**, (5210), pp. 542–545
[16] Acharya, A.K.: 'Image encryption using a new chaos based encryption algorithm'. ACM Int. Conf. Proceeding Series, Rourkela, India, 2011, pp. 577–581
[17] Mondal, B., Mandal, T.: 'A light weight secure image encryption scheme based on chaos & DNA computing', *J. King Saud Univ. - Comput. Inf. Sci.*, 2017, **29**, (4), pp. 499–504
[18] Hermassi, H., Belazi, A., Rhouma, R., *et al.*: 'Security analysis of an image encryption algorithm based on a DNA addition combining with chaotic maps', *Multimed. Tools Appl.*, 2014, **72**, (3), pp. 2211–2224
[19] Ellaway, R.H., Pusic, M. V., Galbraith, R.M., *et al.*: 'Developing the role of big data and analytics in health professional education', *Med. Teach.*, 2014, **36**, (3), pp. 216–222
[20] Church, G.M., Gao, Y., Kosuri, S.: 'Next-generation digital information storage in DNA', *Science*, 2012, **337**, (6102), p. 1628
[21] Banal, J.L., Shepherd, T.R., Berleant, J., *et al.*: 'Random access DNA memory in a scalable, archival file storage system'. bioRxiv, 2020, pp. 1–29
[22] Selvaraj, S., Sundaravaradhan, S.: 'Challenges and opportunities in Iot healthcare systems: a systematic review', *SN Appl. Sci.*, 2020, **2**, (1), p. 139
[23] Dimitrov, D.V.: 'Medical internet of things and big data in healthcare', *Healthc. Inform. Res.*, 2016, **22**, (3), pp. 156–163
[24] Sabry, M., Hashem, M., Nazmy, T.: 'Three reversible data encoding algorithms based on DNA and amino acids' structure', *Int. J. Comput. Appl.*, 2012, **54**, (8), pp. 24–30
[25] Bornholt, J., Lopez, R., Carmean, D.M., *et al.*: 'A DNA-based archival storage system'. Int. Conf. on Architectural Support for Programming Languages and Operating Systems - ASPLOS, Atlanta, USA., April 2016, pp. 637–649
[26] Bancroft, C.: 'Long-term storage of information in DNA', *Science*, 2001, **293**, (5536), p. 1763
[27] Goldman, N., Bertone, P., Chen, S., *et al.*: 'Towards practical, high-capacity, low-maintenance information storage in synthesized DNA', *Nature*, 2013, **494**, (7435), pp. 77–80
[28] Shimanovsky, B., Feng, J., Potkonjak, M.: 'Hiding data in DNA'. Int. Workshop on Information Hiding, Noordwijkerhout, The Netherlands, October 2002, pp. 373–386
[29] Takahashi, C.N., Nguyen, B.H., Strauss, K., *et al.*: 'Demonstration of End-to-End automation of DNA data storage', *Sci. Rep.*, 2019, **9**, (1), pp. 1–5
[30] Organick, L., Chen, Y.J., Dumas Ang, S., *et al.*: 'Probing the physical limits of reliable DNA data retrieval', *Nat. Commun.*, 2020, **11**, (1), pp. 1–7
[31] Qian, L., Winfree, E., Bruck, J.: 'Neural network computation with DNA strand displacement cascades', *Nature*, 2011, **475**, (7356), pp. 368–372
[32] Cherry, K.M., Qian, L.: 'Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks', *Nature*, 2018, **559**, (7714), pp. 370–376
[33] Kim, J., Hopfield, J.J., Winfree, E.: 'Neural network computation by in vitro transcriptional circuits'. Neural Information Processing Systems, Vancouver, Canada, Decemebr 2004, pp. 681–688
[34] Chen, J., Antipov, E., Lemieux, B., *et al.*: 'DNA computing implementing genetic algorithms'. Evolution as Computation DIMACS Workshop, Princeton, USA., January 1999, pp. 39–49
[35] Ren, L., Ding, Y., Ying, H., *et al.*: 'Emergence of self-learning fuzzy systems by a new virus DNA-based evolutionary algorithm', *Int. J. Intell. Syst.*, 2003, **18**, (3), pp. 339–354
[36] Yoshikawa, T., Furuhashi, T., Uchikawa, Y.: 'Effects of combination of DNA coding method with pseudo-bacterial GA'. Proc. of the IEEE Conf. on Evolutionary Computation, ICEC, Indianapolis, USA., April 1997, pp. 285–290
[37] Deaton, R., Murphy, R.C., Rose, J.A., *et al.*: 'DNA based implementation of an evolutionary search for good encodings for DNA computation'. Proc. IEEE Conf. on Evolutionary Computation, ICEC, Indianapolis, USA., April 1997, pp. 267–271
[38] Li, Y., Fang, C., Ouyang, Q.: 'Genetic algorithm in DNA computing: a solution to the maximal clique problem', *Chinese Sci. Bull.*, 2004, **49**, (9), pp. 967–971
[39] Kuurniawan, T.B., Khalid, N.K., Ibrahim, Z., *et al.*: 'Sequence design for direct-proportional length-based DNA computing using population-based ant colony optimization'. Proc. ICCAS-SICE, Fukuoka, Japan, November 2009, pp. 1486–1491
[40] Ibrahim, Z., Kurniawan, T.B., Khalid, N.K., *et al.*: 'Implementation of an ant colony system for DNA sequence optimization', *Artif. Life Robot.*, 2009, **14**, (2), pp. 293–296
[41] Dorigo, M., Gambardella, L.M.: 'Ant colony system: A cooperative learning approach to the traveling salesman problem', *IEEE Trans. Evol. Comput.*, 1997, **1**, (1), pp. 53–66
[42] Wood, D.H.: 'DNA computing capabilities for game theory', *Nat. Comput.*, 2003, **2**, (1), pp. 85–108
[43] Cukras, A.R., Faulhammer, D., Lipton, R.J., *et al.*: 'Chess games: A model for RNA based computation', *Biosystems*, 1999, **52**, (1–3), pp. 35–45
[44] Macdonald, J., Stefanovic, D., Stojanovic, M.N.: 'DNA computers for work and play', *Sci. Am.*, 2008, **299**, (5), pp. 84–91
[45] Challagulla, N.V., Rohatgi, V., Sharma, D., *et al.*: 'Recent developments of nanomaterial applications in additive manufacturing: a brief review', *Curr. Opin. Chem. Eng.*, 2020, **28**, pp. 75–82
[46] Raghupathi, W., Raghupathi, V.: 'Big data analytics in healthcare: promise and potential', *Heal. Inf. Sci. Syst.*, 2014, **2**, (1), p. 3
[47] Kayyali, B., Knott, D., Van, K.S.: 'The big-data revolution in US health care: accelerating value and innovation', *McKinsey Co.*, 2013, **2**, (8), pp. 1–13
[48] Reddy, A.R., Kumar, P.S.: 'Predictive big data analytics in healthcare'. Proc. 2nd Int. Conf. on Computational Intelligence and Communication Technology, Ghaziabad, India, 2016, pp. 623–626
[49] Erlich, Y., Zielinski, D.: 'DNA fountain enables a robust and efficient storage architecture', *Science*, 2017, **355**, (6328), pp. 950–954

7 Appendix

7.1 Example 1. Input data: an adjacency matrix for a graph G(V,E)

7.1.1 Data encoding: The binary equivalent of the adjacency matrix for this graph would be as follows:

011011101100100000111011011111101011001110000010110010110011101100100011100101111010111000000100000000111100000001011000110001000100100111010101100111110011001001011100001100100100011110000100110100000000011011101110100010000001010000101111110100110111010011101110001000101110110011011011001011001000001010011101101011111111111101101011101110010000001000100001111010011011000100111

Use the following table to convert binary string to a nucleotide string: A- 00, C- 01, G-10, T-11

DNA:
CGTGTAGAATGTCTTGGTGCTAAGTAGTATGTCATAACTGC
CTGGTGAACAAAATTAAACCGATACACAGCTCCCGCTTAT
GCAGTGACGCAGATTAAGCGGAAAATCTCTCACAAAGGA
CCTTGGCGTGGCTCTAAGAGTGTATCGTAGTAGAAGGCTC
GGTTTTTTGTCCTCTAGAACACAATTCATCACAT

Fragment1:
GTGAGATGTTTGTGCAAGAGTTGTATACTGCTGTGACAAA
TTAACGATCACGCTCCGTTAGCATGAGCAATTAGCGAAAT
CCTCCAAGGACTTGCGGGCCTAGAGGTACGTGTAAAGCT
CGTTTTTTCCCTAAACCAATCACAC

Fragment2:
CTACGTTACACGAAACAACCTGCGAGATAACGTTATTAGG
GTGTGATT

Fragment3:
TGTTCTTGTGGTCATTTCTCATGTCGCCTGAGTCAGCAAAT
TAACTCGCACGCTAATGGCCGTCAGTACGCCAGCAGGATC
GAGCAAGGAAGGATACCGGATGAGCATGCAGTAGGATCT
ACCAAACTTGATGGTCAAAGTGTG

Fragments after introducing 1% random error

Fragment1Err:
GTGAGATGTTTGTGCAAGAGTTGTATACTGCTGTGACAAA
TTAACGATCACGCTCCGTTAGCATGAGCAATTAGCGAAAT
CCTCCAAGGACTTGCGGGCCTAGAGGTACGTGTAAAGCT
CGTTTTTACCCTAAACCAATCACAC

Fragment2Err:
CTACGTTACACGAAACAACCTGCGAGATAACGTTATTAGG
GTGTGATT

Fragment3Err:
TGTTCTTGTGGTCATTTCTCATGTCGCCTGAGTCAGCAAAT
TAACTCGCACGCTAATGGCCGTCAGTACGCCAGCAGGATC
GAGCAAGGAAGGATACCGGATGAGCATCCAGTAGGATCT
ACCAAACTTGATGGTCAAAGTGTG

7.1.2 Data retrieval: DNA1:
CGTGTAGAATGTCTTGGTGCTAAGTAGTATGTCATAACTGC
CTGGTGAACAAAATTAAACCGATACACAGCTCCCGCTTAT
GCAGTGACGCAGATTAAGCGGAAAATCTCTCACAAAGGA
CCTTGGCGTGGCTCTAAGAGTGTAACGTAGTAGAAGGCTC
GGTTTTTTCACCTCTAGAACACAATTCATCACAT

DNA2:
CGTGTAGAATGTCTTGGTGCTAAGTAGTATGTCATAACTGC
CTGGTGAACAAAATTAAACCGATACACAGCTCCCGCTTAT
GCAGTGACGCAGATTAAGCGGAAAATCTCTCACAAAGGA
CCTTGGCGTGGCTCTAAGAGTGTATCGTAGTAGAAGGCTC
GGTTTTTTGACCTCTAGAACACAATTCATCACAT
DNA3:
CGTGTAGAATGTCTTGGTGCTAAGTAGTATGTCATAACTGC
CTGGTGAACAAAATTAAACCGATACACAGCTCCCGCTTAT
GCAGTGACGCAGATTAAGCGGAAAATCTCTCACAAAGGA
CCTTGGCGTGGCTCTAAGAGTGTATGGTAGTAGAAGGCTC
GGTTTTTTGTCCTCTAGAACACAATTCATCACAT

7.1.3 Final result retrieved: CGTGTAGAATGTCTTGGTGCTAAGTAGTATGTCA
TAACTGCCTGGTGAACAAAATTAAACCGATACACAGCTCC
CGCTTATGCAGTGACGCAGATTAAGCGGAAAATCTCTCAC
AAAGGACCTTGGCGTGGCTCTAAGAGTGTATCGTAGTAGA
AGGCTCGGTTTTTTGTCCTCTAGAACACAATTCATCACAT

7.2 Example 2. Input data: 'following is the original string'

7.2.1 Data encoding: The binary equivalent of the adjacency matrix for this graph would be as follows:
0100011001101111011011000110110001101111011011101101
1001011011100110011100100000011010010111001100100000011
1010001101000011001010010000001101111011100100110100101
1100111011010010110111001100001011011000010000000111001
1011101000111001001101001011011100110011100101110

Use the following table to convert binary string to a nucleotide string:
A- 00, C- 01, G-10, T-11
Corresponding DNA string:
CACGCGTTCGTACGTACGTTCTCTCGGCCGTGCGCTAGAA
CGGCCTATAGAACTCACGGACGCCAGAACGTTCTAGCGG
CCGCTCGGCCGTGCGACCGTAAGAACTATCTCACTAGCGG
CCGTGCGCTAGTG
Fragment1:
ACGGTTGTAGTAGTTTCTGGCGTGGCTGAAGGCTATGAAT
CAGGAGCCGAAGTTTAGGGCGCTGGCGTGGACGTAGAAT
ATTCATAGGGCGTGGCTGTG
Fragment2:
CCCCCCCCCCACCACCCACCCCCCCCACCCCCCA
Fragment3:
CATTGGTGCTGCTGGGAGTTATGTTAGGAATTAGCGGAAG
ACTTCTAAGAATGGGCTTTATAGTTATGTTCATGCGAAGCG
GACGCTTTATGTTAGGTG (Generated using the XOR operation)
Fragments after introducing error:
Frag1Err:
TCGGTTGTAGTAGTTTCTGGCGTGGCTGAAGGCTATGAAT
CAGGAGCCGAAGTTTAGGGCGCTGGCGTGGACGTAGAAT
ATTCATAGGGCGTGGCTGTG
Frag2Err:
CCCCCCCCCCACCACCCACCCCCCCCACCCCCCA
Frag3Err:
CATTGGTGCTGCTGGGTGTTATGTTAGGAATTAGCGGAAG
ACTTCTAAGAATGGGCTTTATAGTTATGTTCATGCGAAGCG
GACGCTTTATGTTAGGTG

7.2.2 Final result retrieved: CACGCGTTCGTACGTACGTTCTCTCGGCCGTGC
GCTAGAACGGCCTATAGAACTCACGGACGCCAGAACGTT
CTAGCGGCCGCTCGGCCGTGCGACCGTAAGAACTATCTCA
CTAGCGGCCGTGCGCTAGTG
Retrieved string: 'Following is the original string'.