On Optimal Family of Codes for DNA Storage

Vijay Dhameliya · Dixita Limbachiya · Madhav Khakhar · Manish K Gupta

Abstract Advancement in the technology has generated immense data which has become a burning issue for data storage experts. To accommodate the data and cope up with demand, computer scientist are striving to produce the improved, dense and reliable data storage medium. The main challenge for the development of optimum data storage medium is enhancement in storage capacity, reliability and security. Various applications of DNA in computing technologies and its dense, stable, reliable nature enticed the researcher to use DNA as storage medium. The idea of using DNA as storage medium has many success stories but the main challenges to deal with are error correction and cost associated with the DNA sequencing and synthesis. In this work, we have developed an efficient technique to encode the data into DNA by using non-linear family of ternary codes. This gives us significant reduction in file size for storing data on DNA from previous developed methods. Using our method one can store 1.15 ExaBytes (EB) of information in one gram of DNA.

Keywords DNA storage · Golay Codes · Non-linear ternary codes · DNA nanotechnology · Biostorage · Synthetic biology · DNA Computing · Error correction codes · Natural data storage · Goldman’s DNA storage.

1 Introduction

Advancement in the technologies like digitization, big data, social networking and IoT (Internet of Things) has produced huge amount of data and this...
explosion of the data has urged the data scientists to strive for the development of better data storage medium. Optical, digital and cloud data storage [10] [4] medium have their own limitations and need to be maintained regularly. While the computer scientists are endeavoursing to develop effective and dense data storage medium, researchers at the other end thought of exploring new medium to safeguard this data. Looking at the potential applications of DNA in various in the fields of computation [22], data security [7] [15] and cryptography [17] [3], researchers thought of employing DNA for data storage. DNA storage is interdisciplinary field that bridge the ideas of different research areas namely data storage, DNA, error correction codes and security. Properties of DNA like scalability, density and long term stability makes it ideal for long term archival of data. A good deal of work has been done to store the data on DNA [9] [8] [23] [6]. But there was no guarantee of error correction in these methods. For reliable transmission and storage of information, error correction codes are inevitable. By employing error correction for DNA data storage, the breakthrough was made by N. Goldman and his team [14] in the year 2013. They developed very efficient and novel approach for storing the data of size 757 KB on DNA and retrieving it back with 100 percent accuracy. They used four folds redundancy to tackle errors and data loss by retrieving correct data from one of the copies of DNA strands. But due to redundancy there was increase in the length of DNA which makes the technique expensive to use DNA for storage at the commercial level. Motivated by DNA computing several researchers have studied DNA codes (set of DNA strings satisfying different combinatorial and thermodynamical constraints) [12] [19] [11] [1] [18].

Work of Goldman [14] and Church [6] has laid a cornerstone for DNA based data storage systems but error correction model for DNA storage is inevitable. Recently model of DNA channel for data storage is proposed by Han Mao Kiah et al., [16]. In this work, we have developed one such error correction scheme (a family of non linear ternary codes) by modifying Goldman’s scheme [14] for the error detection and correction along with improvement in storage capacity to store data on DNA.

The paper is organized as follows. Section II describes the error correction schemes in DNA based storage medium briefly. Section III includes introduction and construction of non linear family of codes and IV discuss about the analysis of class of codes. Algorithm for encoding/decoding, DNA Cloud 2.0 and conclusion is described in section V, VI and VII respectively.

2 Error Correction in DNA Based Information Storage System

The most efficient DNA encoding used till date for DNA based information storage is one bit per base encoding system. Earlier Clelland et al., in 1999 [7] have developed the DNA based stenography technique for encoding the secret messages in DNA. Yachie et al., [25] [24] used alignment based approach to convert the encoded information into hexadecimal value and finally binary values. The last step was to translate the bit data sequence into four multiple
oligonucleotide sequence. This was mapped with the nucleotide base pairs to encode data in DNA. In 2009 Ailenberg et al., [2] proposed an improved Huffman coding based technique to store information on DNA. Rewritable bacterial storage system was introduced which encoded data in recombinant DNA inserted into vector using four number system and 64 codon sequence uniquely mapped to ASCII characters by using mapping table 0=A, 1=T, 2=C and 3=G for the generation of DNA strand [5]. Recently researcher used four bit per two base system along with standard data compression methods and using Huffman coding to compressed data and binary bits converted to nucleotides using four bit per two base mapping [21]. Low Density Parity-Check (LDPC) codes based method was introduced by Aldrin Kay-Yuen Yim et al., [26] to encode data in DNA for improved length of DNA chunks and better method for handling errors used for the DNA data storage. In this approach data was compressed using lossless compression and file was divided into six fragments and each fragment was placed in LDPC error correction code and file was converted to nucleotides by simple base conversion. Most of methods used till date have limitation like without reliable error correction method, size of the data inserted in DNA and cost associated.

In 2013 Goldman used one bit per base system introduced by Church and modifying it by employing the improved base 3 Huffman code (trits 0, 1 and 2) encoding scheme as described in Fig 1. In this computer file was converted to base 3 using Huffman coding and then codes were converted to DNA and introduced redundancy for error correction. It involved four steps shown in Fig 2. Binary digits holding the ASCII codes was converted to base-3 Huffman code that replaces each byte with five or six base-3 digits (trits). Each of trit was encoded with one of the three nucleotides different from the previous one used to avoid homopolymers that cause error in synthesis of DNA. DNA strand was divided into chunks each of length 117 base pair (bp). To include redundancy for error detection and correction, 75 bases for each DNA information chunks were overlapped for four fold redundancy to get the data loss that occurred during synthesis and sequencing DNA. For the data security each redundant chunk was converted to reverse complement of the strand in every alternate chunks. Each DNA chunk was appended with an indexing information bit to determine the location of the data in each chunk and the file from which it is generated. At the end parity check bit was added for error detection. For details reader is referred to [14].

3 Family of Non-Linear Ternary Codes

During the information transmission, redundancy in codes enables the receiver to identify as well as correct certain number of errors that have occurred anywhere in the message without retransmission of message. Goldman used four fold redundancy to correct error of 2 bits (trits) but due to redundancy the length of the DNA used to encode data is very large that make the DNA based storage medium more expensive for any practical application. In this work we
modified Fig 1 by using non-linear ternary error correction codes instead of Huffman codes as referred in Fig 4. To improve the error correction and decreased the length of the DNA and cost associated with it, here we introduced a class of non linear ternary codes for DNA based data storage medium. Any arbitrary computer file can be converted into list of ASCII values ranging from 0 to 255. So we need set of 256 ternary codewords, each corresponding to one value in \{0,..,255\}, to encode any such file into DNA string. For this we need error correction code with distinct 256 codewords and error correction capacity better than 2 bits. From exhaustive search and enumeration of codewords (listed in the table 1) we identified and constructed seven families of non linear ternary codes with the parameters length n, dimension k and minimum distance d \[n,k,d\] that have distinct 256 codewords and error correction capacity t. They are with \[n,k,d\] parameters as \[11,6,5\], \[9,6,3\], \[15,6,7\],
Fig. 2 Stepwise encoding of data in DNA using Goldman approach is explained in detail.

First family of non-linear codes developed are ternary Golay codes \([13]\) which are constructed over fields \(\mathbb{Z}_3\) is perfect error correction code used for different applications in digital communication. Others code design technique which makes use of codewords that resulted in subcode family of ternary Golay codes that has codewords with different \(n, k, d\) parameters. Here one of subcode family with \([11, 6, 5]_3\) is described in details. The ternary Golay code consists of \(3^6 = 729\) codewords of length 11, with minimum hamming distance 5. So it allows receiver to identify 4 trits and correct 2 trits of errors occurred anywhere in any codeword. It is inescapable to have codewords in base-3, because while encoding codewords into DNA we have only three nucleotides different from the last one to avoid homopolymers runs \([14]\).

In this work step b and c of Fig 2 was modified by using ternary Golay codes instead of Huffman codes and error correction without redundancy see Fig 5. Each byte of a computer file is encoded using ASCII to Golay code and stored in the new string. We considered subcode of Golay code (given in the table \([11]\)) consisting of 243 codewords from 729 Golay codewords such that minimum hamming distance between any two codewords is 6, and assigned them to 243 ASCII values having highest probability of occurrences according to frequency used by Goldman described in the table in supplementary file. Remaining 13 ASCII values, were assigned random codewords from remaining set of ternary Golay codewords i.e. these 13 codewords will have minimum hamming distance 5 with other 243 codewords. This ensures that by maximum
likelihood decoding we can correct up to 2 trits flip error. Since there exists only 243 codewords in set of $3^{11}$ codewords with minimum hamming distance 6, it is not possible to construct code consisting of minimum 256 codewords such that length of codewords is 11 and minimum hamming distance more than 5. The file information (i.e. file size and file extension) is encoded using same codewords set and appended at the end of this string, separated by Golay codes of comma and semicolon as separator. Followed by this all the ternary codewords were converted to DNA by using only three nucleotides different from the last one used to encode current bit into nucleotide. The resulting DNA sequence is split into segments of 99 bases as shown in Fig 3 where $t$ is number of basepairs required for the information storage, $\lambda$ is base pairs required to store file index number that can be calculated as file index number (no of file index trits = 2, thus allowing maximum of $3^2 = 9$ files to be distinguished), number of bases requires for chunk index (no of segment index trits $\mu = \lceil \log_3(\text{total no of segments}) \rceil$) and one parity-check trit \[14\] is encoded as non-repeating DNA nucleotides and appended at the end of each segment of length 99 bases.

**Example 1** We choose five computer files used by Goldman \[14\] as a proof of concept for refurbished algorithm for DNA-storage. This File were converted into DNA sequence using software DNA Cloud 2.0. These files constitutes total of 757,051 bytes, each byte was encoded into codeword of length 11 in base-3, picked from ternary Golay code. The resultant ternary sting was encoded into DNA sequences with no allowance for homopolymers (facsimileing the scheme used by Goldman to avoid runs of 2 identical bases because such runs cause errors while sequencing the DNA as in Church et al experiment \[6\]). In nutshell, the five files were stored in 84126 DNA strings, size of each varying from 109 to 112 nucleotides (nt) (depending on number of segments generated from a particular file), resulting in storage of 757,051 bytes of computer data on 9,386,191 nt. This is stupendous result compare to Goldman’s encoding scheme by which the same 757,051 bytes were stored on 153,335 DNA chunks, each of 117 nt, or total of 17,940,195 nt \[14\] (almost twice of the number of nucleotide required by Golay code approach). Note that we have not performed synthesis and sequencing protocols for the encoded data in DNA. Results and
analysis is performed for data embedded DNA sequences generated by the purpose written software.

### 3.1 Error Correction Technique

This tactic of using Golay code facilitates the detection and correction of the write/read error occurred during synthesis/sequencing of DNA, in considerable amount. Each DNA sequence was split into segments and each segment was concatenated with file and segment indexing information to determine the file from which the segment has originated and position at which it resides in the file, while constructing the files back from DNA. Thus codewords constructed here produces optimal error correction restricting the length codewords to 11
Improved error correction approach for DNA data storage using Golay codes is developed here. It depicts the steps of conversion of file into DNA sequence by using Golay code.

in order to achieve optimal storage capacity simultaneously. If there occurs 1 or 2 flip errors during DNA synthesis/sequencing in block of 11 nucleotide, it can be identified as well as corrected. These errors will be transferred to ternary codeword when nucleotides are decoded using Goldman’s table. Since the minimum hamming distance between any two ternary Golay codewords in the table is 5, derived ternary codeword will not match any other codeword in table and will have hamming distance 1 or 2 with the original codeword. Any other error correction code will produce better error correction at the cost of significant reduction in storage capacity.

4 Analysis

Family of non linear codes described here has many advantages over codes used for DNA data storage. It has properties like higher data density, DNA Shannon information, DNA storage cost and code rate. Each of them is described below in brief.
4.1 DNA information density

Amount of data that can be encoded in DNA can be quantified by DNA information density.

**Definition 1** For a given DNA based information storage system DNA information density is total amount of data that can be stored in unit gram of DNA.

At theoretical maximum, one gram of single stranded genetic code can encode 455 EB (exabytes) of information. Goldman achieved information density $2^{2.2}$ PB (petabytes) per gram of DNA. Using DNA Golay codes computationally, we achieved information density for DNA based storage medium as $1.15 \times 10^{20} = 115$ EB (115000 PB). Below is described the derivation of DNA information density

**Remark 1** Consider total information that can be encoded in one gram of DNA is $x$ bytes.

1. Number of nucleotides required to store file is denoted by $I$

   \[ I = 11 \times x + 11 \times 2 + 11 \times \log_{10}(x) \]  

   where $x \times 11$ are nucleotides for $x$ bytes, $2 \times 11$ are nucleotides for 2 separators mentioned in algorithm, $\log_{10}11 \times (x)$ are nucleotides for storing file size on DNA.

2. Number of chunk $C = \lceil I/99 \rceil$, each chunk will have 99 nucleotide of file information

3. Length of chunk $l = \lceil (\log_3(C)) \rceil + 102$, $\lceil \log_3(C) \rceil$ is chunk index information.

4. Number of total nucleotides for storing file on DNA

   \[ \lceil 102 + \log_3(I/99) \rceil \times (x \times 11 + 2 \times 11 + \log_{10}(x) \times 11) / 99 \]  

   Since maximum storage capacity of DNA is 443 (whatever Exabytes) we can consider $\log_{10}(x) = 20$.

5. 1 gm of DNA consist of $182 \times 10^{19}$ nucleotides so for calculating number of bytes that can be stored on 1 gm of DNA :

   \[ 182 \times 10^{19} = \frac{(102 + \log_3(\frac{1}{99})) \times (x \times 11 + 2 \times 11 + \log_{10}(x) \times 11)}{99} \]  

   \[ x = 1.15 \times 10^{20} \text{ bytes (115 Exabytes).} \]
4.2 DNA Shannon Information

To quantify the data stored in DNA, we need a measure called DNA Shannon information that states the number of nucleotides used to store specific amount of data.

**Definition 2** DNA Shannon information can be defined as required amount of nucleotides to store file of unit size (byte).

For the proof of concept we converted five files used by Goldman to DNA using DNA Golay codes. Fig 6 it can be observed that DNA Golay codes has higher information than Goldman approach. DNA required to store data using code design described here is much less than Goldman approach. Shannon information for DNA based storage medium by Goldman approach is 0.042 bytes per base. DNA Shannon information using DNA Golay codes is calculated here:

\[
\text{DNA Shannon information} = \frac{\text{Size of files (Bytes)}}{(\text{Number of chunks} \times \text{Chunk size})}
\]

For given file of size 757051 bytes, number of chunks is 84126 and chunk size is 112; DNA shannon information for DNA Golay codes approach is 0.082 bytes per base. One can observe that DNA Shannon information for Goldman approach is less than DNA Golay codes (ie. 0.042 bytes per base < 0.082 bytes per base.).

4.3 DNA storage cost

Cost of DNA synthesis and sequencing is major limitation of DNA storage medium. Considering synthesis cost $0.05 per base [14], total cost per MB for different file size (bytes) is plotted here. Using DNA Golay code, required amount of the DNA is decreased which resulted into decreased in the cost. Fig 7 shows the cost to store data using various DNA based information encoding techniques. With the increase in the amount of data to be stored in DNA there is negligible change in cost used to store unit MB associated with DNA based storage using non-linear family of codes develop here. On the other side, cost for the DNA increases with the increases in file size using Goldman method. One can observe using DNA Golay code DNA as storage medium becomes cheaper for long term archival.

4.4 Tradeoff for code rate

The storage capacity of our encoding scheme is 0.73 bits per base by encoding 8 bits into 11 DNA bases per byte (i.e. 8/11 = 0.73 bits per base). It can be improved further to 0.89 by using 9 base per byte (i.e. 8/9 = 0.89 bits per base).
Fig. 6 The graph shows that number of nucleotide required to store the data by using Goldman and Golay DNA codes. Observe that data density of Golay DNA codes is greater than Goldman.

base) by encoding each byte using code [9, 6, 3], at the cost of reduction in error detection capacity to 2 trits and correction capacity to 1 trits per codeword. Fig 8 shows tradeoff for each family of codewords develop along with the t error correction. It can be observed that error correction capacity t for each code is increased linearly with increase in length of the code n. But with increase in error correction capacity and length of code there is declination in code rate. For reliable data storage, code rate is optimal with [11, 6, 5] code, if one choose code with [9, 6, 3] there is decrease in code rate hence there is open challenge for the construction of codes with optimal code rate for DNA based storage medium. Hence one can conclude that code of family [11, 6, 5] is optimal code.
Fig. 7 Comparison of cost of DNA synthesis and sequencing using Goldman and family of Golay codes used in DNA based information storage. Number of nucleotides required to store the data. The graph shows that number of nucleotide required to store the data by using Goldman and Golay DNA codes and data density of Golay DNA codes is greater than Goldman.

5 Algorithm for Encoding and Decoding Data Files

The sequential procedure for encoding and decoding data from file into DNA and vice versa is described in the algorithm 1 and 2 respectively. For code length 11, encoding and decoding is described below.

6 DNA Cloud 2.0

DNACloud 1.0 was tool based on Goldman’s algorithm for converting any computer file into DNA sequence [20]. The next version of the software i.e. DNACloud 2.0, includes the feature to encode/decode the computer files into DNA using DNA Golay codes described in this paper. DNACloud 2.0 is featured with data compression which allows user to convert the file into DNA after compressing it. The file type specific standard compression techniques are used for compression i.e. bz2 for text data files and FFmpeg for multimedia files.
Algorithm 1 Algorithm for Encoding
Input: Any arbitrary computer file
Output: DNA sequences in which data is encoded

1.1 \( |S| \) = number of elements in string S.
1.2 Any arbitrary computer file is read byte wise and each byte is converted into corresponding ASCII value which is stored in list \( S_0 \) where \( S_0 \in \sum_{255} = \{0,1,...,255\} \).
1.3 Each element from \( S_0 \) is encoded using given code table (in supplementary file), converting it to base-3 string of length 11, and stored in string \( S_1 \) where \( S_1 \in Z_3^* = \{0,1,2\} \) and \( |S_1| \equiv 0 \pmod{11} \).
1.4 \( S_2 \) and \( S_4 \) are codes of characters comma and colon respectively. File extension of input file is encoded into Golay code and stored in string \( S_3 \).
1.5 \( N = |S_1| \). String N is encoded into code and stored in \( S_5 \).
1.6 \( n = |S_1| + |S_2| + |S_3| + |S_4| + |S_5| \).
1.7 \( S_6 \in \sum_0^* = \{0\} \) where \( (n + |S_6|) \equiv 0 \pmod{99} \).
1.8 \( S_7 = S_1.S_2.S_3.S_4.S_6.S_5 \).
1.9 \( S_7 \) is encoded into a DNA string \( S_8 \) where \( S_8 \in \sum_{DNA} = \{A,C,T,G\} \) with no repeated nucleotides using the scheme used by Goldman (The first trit of \( S_7 \) is coded using the 'A' row of the table).
1.10 \( S_8 \) is split into segments of length 99, each such segment is called chunk and stored in list \( L \) in their original order.
1.11 \( \mu = \lfloor \log_3(|S_8|/99) \rfloor \), \( \mu \) is number of trits required to store index number of chunk.
1.12 \( i \) is chunk index in base-3. ID is 2-trit string identifying the original file. P is parity trit computed as the \( \text{sum} \pmod{3} \) of the odd-positioned trits in ID and \( i \); i.e. \( ID(1) + i(1) + i(3) + i(5) \) likewise.
1.13 IX = ID.I.P and IX is encoded into DNA in the same way as in step 9 taking last character of chunk from \( L \) as previous character for conversion. \( K \) is the list of chunks such that \( K(i) = L(i).IX(i) \).
1.14 The chunks from \( K \) can be synthesized as DNA oligonucleotides, stored for long term and sequenced.

Algorithm 2 Algorithm for Decoding
Input: DNA sequences in which data is encoded.
Output: Original computer file

2.1 \( L \) is list of DNA sequences of length n.
2.2 \( S = L(i) \ [102:n-1] \), the string of nucleotides starting at position 102 and ending at \( n-1 \) in \( i^{th} \) sequence from \( L \), is stored in \( S \). The DNA string \( S \) is decoded into ternary string using nucleotide conversion table described by Goldman and stored in \( i.k = \text{Decimal}(i) \).
2.3 DNA sequences are read from \( D \) in increasing order of their corresponding keys and concatenated to form DNA string \( S_1 \).
2.4 Decode \( S_1 \) using nucleotide conversion table to ternary string \( S_2 \).
2.5 Blocks of 11 trits are read from \( S_2 \) and converted into ASCII values using Golay Table. The characters corresponding to these ascii values are stored in string \( S_3 \). If the sequence of trits is absent in the table, it is decoded using error correction technique.
2.6 \( S_3 \) is split based on last occurring comma into string \( S_4 \) (Original file content) and \( S_5 \) (file information). File extension is extracted from \( S_5 \) and stored in string \( S_6 \).
2.6 Decoded file is generated using \( S_4 \) and \( S_6 \).
Fig. 8 Tradeoff between code rate and length of code is plotted. It shows the optimal code with length 12, error correction 2 has code rate 0.5

7 Conclusion

To develop reliable and secured DNA based information storage system, we present new non-linear family of DNA codes. With the implementation of the ternary Golay codes, we are able to correct 2 flip errors per block of 11 nucleotides along with much improvement in storage capacity and error correction.

Looking at the algorithm implemented by Goldman and comparing it with the approach implemented in this work, we obtained significant difference in terms of data storage density and length of DNA required. Comparison of the DNA sequences used for encoding and decoding of the data into DNA by both the approaches is described in Table 2. Though DNA data storage has many limitations like cost and information retrieval; its higher density, long term storage make it better choice for long term data archival. The data reading and writing in DNA is not as easy as current day digital storage media but advent in the synthesis and sequencing of DNA is a success ladder to make this natural hard drive as a commercial product. Considering error correction and amelioration in information capacity for DNA data storage as major challenge, here we have derived an approach using ternary Golay
codes for adequate error correction and sizeable information capacity. There is around 50 percent of reduction in the length of DNA required to store any computer data compared to the Goldman’s approach which points to the cost reduction in DNA synthesis and sequencing. In nutshell, encoding approach for the data storage in DNA described here has better error correction scheme, high storage density and is cost effective. We hope that our work encourages coding theorist to look into this emerging area and search for best possible DNA codes for data storage applications.

References

1. Abualrub, T., Ghrayeb, A., Zeng, X.N.: Construction of cyclic codes over for DNA computing. Journal of the Franklin Institute 343(45), 448–457 (2006). DOI http://dx.doi.org/10.1016/j.jfranklin.2006.02.009. URL http://www.sciencedirect.com/science/article/pii/S0016003206000330. Modeling, Simulation and Applied Optimization
2. Ailenberg, M., Rotstein, O.D.: An improved huffman coding method for archiving text, images, and music characters in DNA. Biotechniques 47(3), 747 (2009)
3. Arita, M., Ohashi, Y.: Secret signatures inside genomic DNA. Biotechnology progress 20(5), 1605–1607 (2004)
4. Bassoli, R., Marques, H., Rodriguez, J., Shum, K., Tafazoli, R.: Network coding theory: A survey. Communications Surveys Tutorials, IEEE 15(4), 1950–1978 (2013). DOI 10.1109/SURV.2013.013013.00104
5. Bonnet, J., Subsoontorn, P., Endy, D.: Rewritable digital data storage in live cells via engineered control of recombination directionality. Proceedings of the National Academy of Sciences 109(23), 8884–8889 (2012)
6. Church, G.M., Gao, Y., Kosuri, S.: Next-generation digital information storage in DNA. Science 337(6102), 1628–1628 (2012)
7. Clelland, C.T., Risca, V., Bancroft, C.: Hiding messages in DNA microdots. Nature 399(6736), 533–534 (1999)
8. Cox, J.P.: Long-term data storage in DNA. TRENDS in Biotechnology 19(7), 247–250 (2001)
9. Davis, J.: Microvenus. Art Journal 55(1), 70–74 (1996)
10. Dimakis, A.G., Godfrey, P.B., Wu, Y., Wainwright, M.J., Ramchandran, K.: Network coding for distributed storage systems. Information Theory, IEEE Transactions on 56(9), 4539–4551 (2010)
11. Gaborit, P., King, O.D.: Linear constructions for DNA codes. Theoretical Computer Science 334(1-3), 99–113 (2005)
12. Garzon, M.H., Deaton, R.J.: Codeword design and information encoding in DNA ensembles. Natural Computing 3(3), 253–292 (2004)
13. Golay, M.J.: Notes on digital coding (1949)
14. Goldman, N., Bertone, P., Chen, S., Dessimot, C., LeProust, E.M., Sipos, B., Birney, E.: Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. Nature (2013)
15. Jain, S., Chaudhary, H., Bhatnagar, V.: An information security–based literature survey and classification framework of data storage in DNA. International Journal of Networking and Virtual Organisations 13(2), 176–201 (2013)
16. Kah, H.M., Pulko, G.J., Milenkovic, O.: Codes for DNA storage channels. CoRR abs/1410.8837 (2014). URL http://arxiv.org/abs/1410.8837
17. Leier, A., Richter, C., Banahaf, W., Rauhe, H.: Cryptography with DNA binary strands. BioSystems 57(1), 13–22 (2000)
18. Milenkovic, O., Kashyap, N.: On the design of codes for DNA computing. In: Coding and Cryptography, pp. 100–119. Springer (2006)
19. Milenkovic, O., Vasic, B.: Information theory and coding problems in genetics. In: Information Theory Workshop, 2004. IEEE, pp. 60-65. IEEE (2004)
20. Shah, S., Limbachiya, D., Gupta, M.K.: DNAcloud: A tool for storing big data on DNA. Presented at the 11th Annual Conference on Foundations of Nanoscience: Self-Assembled Architectures and Devices (FNANO14) (2014)
21. Vishwakarma, R., Amiri, N.: High density data storage in DNA using an efficient message encoding scheme. International Journal of Information Technology Convergence & Services 2(2) (2012)
22. Watada, J., binti Abu Bakar, R.: DNA computing and its applications. In: Intelligent Systems Design and Applications, 2008. ISDA ’08. Eighth International Conference on, vol. 2, pp. 288–294 (2008). DOI 10.1109/ISDA.2008.362
23. Wong, P.C., Wong, K.K., Foote, H.: Organic data memory using the DNA approach. Communications of the ACM 46(1), 95–98 (2003)
24. Yachie, N., Ohashi, Y., Tomita, M.: Stabilizing synthetic data in the dna of living organisms. Systems and synthetic biology 2(1-2), 19–25 (2008)
25. Yachie, N., Sekiyama, K., Sugahara, J., Ohashi, Y., Tomita, M.: Alignment-based approach for durable data storage into living organisms. Biotechnology progress 23(2), 501–505 (2007)
26. Yim, A.K.Y., Yu, A.C.S., Li, J.W., Wong, A.I.C., Loo, J.F., Chan, K.M., Kong, S., Yip, K.Y., Chan, T.F.: The essential component in dna-based information storage system: robust error-tolerating module. Frontiers in bioengineering and biotechnology 2 (2014)
Table 1: Codewords from subcode of Ternary Golay Code i.e. \([11,6,5]_3\) assigned to 256 ASCII values is given in the table.

| ASCII Values | Golay codes | Weight | ASCII values | Golay codes | Weight |
|--------------|-------------|--------|--------------|-------------|--------|
| 86           | 00002111102 | 6      | 170          | 00001222101 | 6      |
| 92           | 0002112020  | 6      | 138          | 00001101111 | 6      |
| 93           | 00201010122 | 6      | 190          | 00201121102 | 6      |
| 94           | 00220100121 | 6      | 149          | 00101011111 | 6      |
| 217          | 0011211212  | 9      | 20           | 00110220121 | 9      |
| 229          | 02010101220 | 6      | 255          | 02001210010 | 6      |
| 255          | 0201222020  | 6      | 26           | 02010010010 | 6      |
| 82           | 021212000110| 6      | 75           | 02111111012 | 9      |
| 191          | 0202100121  | 6      | 88           | 0222001022  | 6      |
| 130          | 0211202000  | 6      | 75           | 02110010202 | 5      |
| 241          | 02112020121 | 6      | 75           | 02121221102 | 9      |
| 102          | 01221201212 | 9      | 171          | 02121221111 | 9      |
| 146          | 01000012221 | 9      | 72           | 01021012010 | 6      |
| 24          | 01220021010 | 6      | 106          | 01212021222 | 9      |
| 137          | 01202211111 | 6      | 73           | 021012022010| 6     |
| 181          | 0111222201  | 9      | 175          | 01120002220 | 6      |
| 140          | 01112201112 | 7      | 17          | 01002212002 | 6      |
| 240          | 20121212212 | 6      | 214          | 02120010121 | 5      |
| 245          | 01212200101 | 6      | 168          | 02110121101 | 6      |
| 112          | 01022010010 | 6      | 90           | 02220012211 | 6      |
| 248          | 00012100210 | 6      | 194          | 020101201102| 6     |
| 79           | 2001002211  | 6      | 195          | 02001222110 | 6      |
| 165          | 20221111101 | 6      | 245          | 02011122222 | 9     |
| 38           | 20200021111 | 6      | 131          | 02010112201 | 6      |
| 95           | 22122221110 | 6      | 43           | 22100090012 | 6      |
| 246          | 01212212011 | 6      | 124          | 02120221201 | 9      |
| 136          | 02212100100 | 6      | 107          | 02001222202 | 6      |
| 153          | 22201221212 | 6      | 148          | 22200920020 | 6      |
| 54           | 22121201212 | 6      | 241          | 22201011222 | 9      |
| 178          | 2211212021  | 6      | 159          | 22210010010 | 6      |
| 48           | 21112212101 | 9      | 90           | 21102212022 | 9      |
| 39           | 21121120011 | 9      | 219          | 21102112110 | 9      |
| 172          | 21121211221 | 13     | 14           | 21011222220 | 6      |
| 160          | 22121210010 | 6      | 36           | 22122012211 | 6      |
| 225          | 21121011211 | 6      | 129          | 21101211110 | 9      |
| 35           | 22001010002 | 6      | 93           | 21221022120 | 9      |
| 56           | 22221012120 | 6      | 158          | 02121012211 | 9      |
| 143          | 02020101210 | 6      | 123          | 02010102002 | 6      |
| 111          | 0212122221  | 9      | 103          | 02010120120 | 6      |
| 65           | 01110001010 | 6      | 249          | 01000012221 | 6      |
| 226          | 01210221111 | 6      | 144          | 01220020101 | 6      |
| 128          | 01001202011 | 6      | 135          | 02001222020 | 6      |
| 207          | 00022012112 | 6      | 77           | 01021202000 | 6      |
| 192          | 12221010000 | 6      | 186          | 12220121202 | 9      |
| 118          | 12211001121 | 6      | 246          | 12221212020 | 9      |
| 206          | 02021202122 | 6      | 184          | 12122012212 | 9      |
| 237          | 12112200101 | 9      | 188          | 12110212110 | 9      |
| 210          | 01210110102 | 6      | 135          | 12100220001 | 6      |
| 530          | 01201211221 | 6      | 22           | 12101201001 | 6      |
| 3            | 12000110221 | 6      | 46           | 12001221221 | 9      |
| 176          | 11202000921 | 6      | 23           | 11201111220 | 9      |
| 187          | 12211011121 | 6      | 244          | 11200002200 | 6     |
| 235          | 11101202211 | 6      | 244          | 11100122101 | 6      |
| 200          | 11122000002 | 6      | 221          | 11220120121 | 9      |
| 198          | 1112201120 | 6      | 193          | 11002121000 | 6      |
| 182          | 02001221111 | 6      | 122          | 02120212010 | 9      |
| 121          | 00021012100 | 5      | 7            | 00022100211 | 5      |

On Optimal Family of Codes for DNA Storage
Table 2  Executive Summary of data encoded using Goldman approach and DNA Golay codes

| Original File name | File size | Bytes | Chunk Size | # of Chunks | # of nucleotides | Chunk Size | # of chunks | # of nucleotides |
|-------------------|-----------|-------|------------|-------------|------------------|------------|-------------|------------------|
| EBL.jp2           | 179.9 KB  | 184264| 117        | 37423       | 4378491          | 112        | 20476       | 2293312          |
| MLK_excerpt_VBR₄₅ – 85.mp3 | 164.6 KB  | 168539| 117        | 34164       | 3997188          | 111        | 18728       | 2078808          |
| View_huff3.cd.new | 15.3 KB   | 15646 | 117        | 3163        | 370071           | 109        | 1740        | 189660           |
| watsoncrick.pdf   | 274.3 KB  | 280864| 117        | 56911       | 6658587          | 112        | 31209       | 3495408          |
| wssnt10.txt       | 105.2 KB  | 107738| 117        | 21650       | 2533050          | 111        | 11973       | 1329083          |
| **Total**         | **739.3 KB** | **757051** | **117** | **153335** | **17937387**     | **112** | **84126** | **9386191**      |