

Provided by the author(s) and University College Dublin Library in accordance with publisher policies. Please cite the published version when available.

Title	Performance of DNA data embedding algorithms under substitution mutations
Author(s)	Haughton, David; Balado, Félix
Publication Date	2010-12
This item's record/more information	http://hdl.handle.net/10197/2727
Rights	Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Downloaded 2012-05-16T20:44:28Z

Some rights reserved. For more information, please see the item record link above.



Performance of DNA Data Embedding Algorithms under Substitution Mutations

David Haughton and Félix Balado
School of Computer Science and Informatics
University College Dublin
Dublin, Ireland
david.haughton@ucdconnect.ie, felix@ucd.ie

Abstract—DNA data embedding is a relatively recent area which aims at embedding arbitrary information in deoxyribonucleic acid (DNA) strands. One interesting application of DNA data embedding can be tracing pathways of genetic material in novel ways. This paper explores the decoding performance of several DNA data embedding algorithms proposed in the literature, which are also briefly reviewed. DNA may undergo random mutations, which can cause errors at the decoding stage of such algorithms. Although some proposed methods do account for such errors, decoding performance under mutations has not been previously studied in general. The empirical performance comparison that we provide here allows to fairly compare a number of DNA data embedding algorithms under mutations for the first time. The evaluation is undertaken *in silico* by means of Monte Carlo simulations. Additionally, we propose two new DNA data embedding algorithms with good robustness properties.

Keywords-DNA Data Embedding; Decoding Performance; DNA watermarking

I. INTRODUCTION

In recent years the deoxyribonucleic acid (DNA) molecule has become a novel form of data storage with the proposal by various authors to use this intrinsically digital media for embedding nongenetic information in a highly compact way [1]. Apart from storage, another potential application of DNA data embedding is the field of data security, in particular watermarking and steganography. In these areas the goal is to conceal information in DNA, for instance in order to assert intellectual property (IP) rights. A conspicuous use of DNA watermarking recently hit the news worldwide, when Craig Venter's group *watermarked* the genome of an artificial bacteria [2]. An interesting application of DNA data embedding in biological research is fingerprinting, in order to trace genetic material without altering its biological expression [3]. For instance, it has been recently proposed that DNA data embedding may be used to track organisms which may pose a biological threat [4], such as viruses. This would impose a level of accountability on institutions dealing with such material. If an organism is found to exist outside an authorised environment the responsible party can be identified by genome sequencing.

DNA may undergo mutations, which can also negatively affect any information embedded through a data embedding method. For this reason DNA data embedding can be seen

as an instance of digital communications under a noisy channel. The decoder will have to counteract such errors, and some methods will be more efficient than others in doing so. Although a number of DNA data embedding methods have been proposed over the last ten years, a performance comparison of these methods is largely lacking. Here we evaluate the decoding performance of most existing DNA data embedding methods under substitution mutations (point mutations). Furthermore, decoding under errors is not even considered in most prior art, and for this reason we undertake optimum maximum likelihood (ML) decoding where feasible. To the best of our knowledge, the only authors who have attempted a performance evaluation of their method are Yachie *et al* [5].

A. Types of DNA Data Embedding

DNA is fundamentally a digital unidimensional medium, whose alphabet is formed by four nucleotides (bases): Adenine (A), Cytosine (C), Thymine (T), and Guanine (G). Therefore a DNA strand is completely analogous to a sequence of 4-ary information symbols in digital communications. However due to certain biological constraints in the structure of DNA, it is not always possible to build arbitrary DNA sequences in order to encode information. This is because there are two basic types of DNA:

- Noncoding DNA (ncDNA) sequences are regions of a genome which do not encode proteins. The working hypothesis for ncDNA data embedding usually is that these regions can be freely altered or appended without affecting the biology of the host organism. This assertion has to be strongly qualified, since some ncDNA regions do affect the expression of genes (promoters, pseudogenes). The identification of suitable regions is out of the scope of this paper, but as shown by several authors [1], [5] in *in vivo* experiments, such regions can be identified and altered without negatively impacting the organism. Also, isothermality constraints are sometimes placed on ncDNA data embedding methods. This type of constraint means that the C-G bases must be in a certain fixed proportion with the A-T bases, in order to speed up bonding of complementary strands (hybridisation).

- Coding DNA (cDNA) sequences are regions of a genome which encode proteins. In these regions (genes), triplets of bases, called codons, can be translated into amino acids, which are then sequentially assembled to yield proteins. There are 4^3 codons but only 20 amino acids (plus one stop symbol indicating the end of the transcription), which are mapped to codons according to the so-called *genetic code* (see Table II). In cDNA data embedding the fundamental constraint is that any modification of a sequence to embed information must preserve its translation to proteins. This is theoretically possible by exploiting the redundancy of the genetic code. In practice, further constraints must often be taken into account, such as the frequency of individual codons (codon usage). This issue is not dealt with in this paper and as such results presented here represent an upper bound to the achievable performance.

B. Notation and Framework

In the following, when the same Roman letter appears in uppercase and lowercase (e.g., X, x), the former refers to a random variable and the latter to a realisation of it. Calligraphic letters indicate sets (\mathcal{X}), and $|\mathcal{X}|$ is the cardinality of \mathcal{X} . Bold symbols indicate row vectors, $\mathbf{x} = [x_1, \dots, x_n]$. The probability mass function (pmf) of X is denoted by $\Pr(X = x)$, or just $\Pr(x)$ if there is no ambiguity about the random variable. The Hamming distance between two equal length vectors \mathbf{x} and \mathbf{y} , which yields the number of different same index symbols, is denoted by $d_H(\mathbf{x}, \mathbf{y})$. The edit distance is given by $d_e(\mathbf{x}, \mathbf{y})$. This distance metric represents the minimum number of operations required to transform a binary string into another. If \mathbf{x} and \mathbf{y} only differ in symbol switches then the two distance metrics are equivalent, that is, $d_e(\mathbf{x}, \mathbf{y}) = d_H(\mathbf{x}, \mathbf{y})$.

We denote $\mathcal{X} \triangleq \{A, C, T, G\}$ as the set of bases, whose cardinality is $|\mathcal{X}| = 4$. A DNA sequence amounts to a vector $\mathbf{x} = [x_1, x_2, \dots, x_n]$ with $x_i \in \mathcal{X}$. We will denote an information carrying DNA sequence by $\mathbf{y} = f(\mathbf{x}, m)$, where $y_i \in \mathcal{X}$ and the message elements are $m \in \mathcal{M} \triangleq \{0, 1, \dots, l-1\}$. Except otherwise indicated, we will assume that all messages are equally likely.

According to our discussion in Section I, with ncDNA we will assume that \mathbf{y} replaces a segment \mathbf{x} of the same length, or that it is appended to it. A cDNA segment of a genome can be expressed as a sequence of codons $\mathbf{x}^c = [x_1, \dots, x_n]$, with $x_i \in \mathcal{X}^3$. Each codon can be translated into a unique amino acid (or stop symbol) $x'_i \triangleq \alpha(x_i)$, with $x_i \in \mathcal{X}^3$ and $|\mathcal{X}'| = 21$, and $\alpha(x^c)$ is the amino acid sequence corresponding to x^c , that is, its primary structure. The subset of codons associated with amino acid x' is denoted as $\mathcal{S}_{x'} \triangleq \{\mathbf{x} \in \mathcal{X}^3 | \alpha(\mathbf{x}) = x'\}$, which is defined by the genetic code shown in Table II. A cDNA data embedding method using a host strand \mathbf{x}^c has to guarantee the constraint

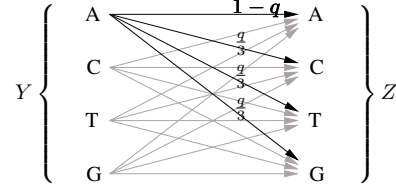


Figure 1: Base substitution mutation channel, with transition probabilities.

$\alpha(\mathbf{y}^c) = \alpha(\mathbf{x}^c)$. Finally note that $|\mathcal{S}_{x'}|$ is the number of synonymous codons that translate into x' , which we call the *multiplicity* of this amino acid.

For ncDNA, a sequence $\mathbf{y}^{(m)}$ corresponding to message m can be called a *codeword* of the method. The length in bases of codeword $\mathbf{y}^{(m)}$ will be denoted as λ_m . If $\lambda_m = \lambda$ for all m , then the *embedding rate* of the method is $R = \frac{1}{\lambda} \log_2 l$ bits/base. In some methods not all codewords have equal length, that is, the embedding rate is variable and depending on the message. With equally likely messages the average embedding rate is $R = \frac{l}{\sum_{m=1}^l \lambda_m} \log_2 l$ bits/base.

When using cDNA we cannot write codewords simply in terms of the information-carrying sequence \mathbf{y} because of the aforementioned genetic constraint. This constraint also causes the embedding rate to be variable, and dependent on the host sequence \mathbf{x} , and so it can only be given as an average. Sometimes it is more convenient to give the embedding rate R in bits/codon when using cDNA. Obviously the rate in bits/base is the rate in bits/codon divided by three.

In order to study the decoding performance of the methods we will assume that \mathbf{y} undergoes a probabilistic substitution mutation channel with output \mathbf{z} . This channel, which is depicted in Figure 1, is just a 4-ary symmetric channel with transition probabilities $\Pr(Z = v | Y = v) = 1 - q$ and $\Pr(Z = w | Y = v) = q/3$, with $v \neq w \in \mathcal{X}$. Therefore q is the probability of base mutation, or base mutation rate. We assume that mutations affecting different bases are mutually independent, that is, a memoryless channel, which can always be approximated in practice by means of a large pseudorandom interleaver shared by encoder and decoder. The performance measurement will be the bit error probability (P_b) after decoding which is an estimate of the message \hat{m} from \mathbf{z} .

C. Maximum Likelihood Decoding

As we already mentioned, many proposed DNA data embedding methods are ambiguous about how to tackle decoding in the presence of mutations. In those cases, we will undertake maximum likelihood (ML) decoding, which is optimal for equally likely messages. The ML decoder makes the decoding decision \hat{m} that maximises the *a posteriori* probability that a received sequence \mathbf{z} corresponds to that

message, that is

$$\hat{m} = \arg \max_{m \in \mathcal{M}} \Pr(\mathbf{Z} = \mathbf{z} | M = m). \quad (1)$$

With mutually independent mutations, the probability required in (1) can be written as $\Pr(\mathbf{z}|m) = \prod_{k=1}^n \Pr(z_k|m)$. The elemental conditional probabilities in this expression are $\Pr(z_k|m) = \Pr(z_k|y_k^{(m)})$, that is, the transition probabilities of the channel model. It is important to note that this ML decoder requires the reading frame for the DNA sequence to be correctly aligned. For this reason, this approach is not suited to decoding under insertion and deletion (*indel*) mutations—which are not studied here—without prior synchronisation (realignment).

II. DESCRIPTION OF THE ALGORITHMS

In this section we review in some detail the most relevant algorithms proposed in the literature for embedding information in DNA. Some of them have been specifically created for DNA data embedding, while others are modifications of methods from related areas, such as source compression. Furthermore some have been successfully implemented using living organisms, that is, *in vivo*, whereas others have only been tried *in vitro* or *in silico*.

Our goal in this section is to set all of these methods on an equal footing, so that fair comparisons are possible. For this reason we will obtain the embedding rate R (in bits/base or bits/codon) for all of these methods, since when two methods have equal decoding performance for the same q the one with a higher R will be superior. Note that it is possible to apply error-correction coding to a message before embedding it. If the error-correction code has a rate r_{ecc} , the overall rate will be $R_o = r_{\text{ecc}} \cdot R$.

A. ncDNA Data Embedding Algorithms

Smith *et al.* have proposed the Huffman code, the comma code and the alternating code for ncDNA data embedding [6]. Another relevant method is DNA-Crypt, by Heider and Barnekow [3].

1) *The Huffman code:* This is a source compression code. If the likelihood of message $i \in \mathcal{M}$ is given by $\Pr(M = i)$, with $\sum_{i=1}^l \Pr(M = i) = 1$, the Huffman code assigns codewords to symbols in such a way that for any two symbols $i, j \in \mathcal{M}$ it holds that $\lambda_i \leq \lambda_j$ iff $\Pr(M = i) \geq \Pr(M = j)$, that is, the most likely symbols are assigned the smallest length codewords. The average length of the encoding is approximately $H(M)$ bits/message symbol, where $H(M)$ is the entropy of the message, from which $R = 2H(M)$ bits/base. Without mutations, the Huffman code is optimal if all the likelihoods are of the form $\Pr(M = i) = 2^{-n_i}$ for some $n_i \in \mathbb{N}$.

Unlike the rest of ncDNA algorithms described in this section, Huffman coding produces variable length codewords. This may be a source of error even for low mutation rates: if the codeword $\mathbf{y}^{(m)}$ with length λ becomes $\mathbf{z}^{(m)}$ with

length $\lambda' \neq \lambda$ after mutation, then the rest of the sequence can become undecodable. This is because in this situation a substitution mutation offsets the reading frame of the decoder, and essentially causes the rest of the message to be lost¹.

2) *The comma code:* The comma code [6] repeats the base G every six bases. Codewords in the comma code are of the form $\mathbf{y} = [\text{GH}_1\text{H}_2\text{H}_3\text{H}_4\text{H}_5]$, with $\text{H}_i \in \{\text{A}, \text{T}, \text{C}\}$, and for every codeword it must hold that $d_H(\mathbf{y}, [\text{GCCCCC}]) = 3$. This means that between two G's there must be a ratio of three A or T bases to two C bases. This results in $\binom{5}{2}2^3 = 80$ possible codeword sequences. The embedding rate is therefore given by

$$R = \frac{1}{6} \log_2 80 = 1.0537 \text{ bits/base}. \quad (2)$$

The structure of the sequence produced enables the detection of some mutations. 83% of substitution mutations that can occur in a codeword result in the formation of an unrecognisable codeword [6]. The authors do not propose a decoding algorithm under mutations so we will use ML decoding.

This code features two advantages: 1) the periodic sequence of G works as a pilot resynchronisation sequence in the presence of *indel* mutations; and 2) it is isothermal.

3) *The alternating code:* DNA bases belong to two classes: purines $\mathcal{R} \triangleq \{\text{A}, \text{G}\}$ and pyrimidines $\mathcal{Y} \triangleq \{\text{T}, \text{C}\}$. The alternating code constructs a 6-bases long DNA sequence which is composed of purines followed by pyrimidines repeatedly [6]. For example, $\mathbf{y} = [\text{Y}_1\text{R}_1\text{Y}_2\text{R}_2\text{Y}_3\text{R}_3]$ or $\mathbf{y} = [\text{Y}_1\text{Y}_2\text{Y}_3\text{R}_1\text{R}_2\text{R}_3]$, with $\text{Y}_i \in \mathcal{Y}$, $\text{R}_i \in \mathcal{R}$, would be acceptable codeword patterns for an alternating code. In any case, since the use of pyrimidines and purines yields a binary alphabet, we have that the rate of this method is

$$R = \frac{1}{6} \log_2 2^6 = 1 \text{ bit/base}. \quad (3)$$

The code structure allows for the detection of some mutations, however less than the comma code. 67% of substitution mutations that can occur in a codeword result in the formation of an unrecognisable codeword [6], and so ML decoding will be also used. This code is also isothermal.

4) *DNA-Crypt:* Heider and Barnekow have proposed and implemented *in vivo* the DNA-Crypt algorithm for embedding data in both ncDNA and cDNA [3]. The ncDNA version of DNA-Crypt is the trivial code that assigns 2 bits/base (since $\log_2 |\mathcal{X}| = 2$). The embedding rate is obviously $R = 2$ bits/base. This algorithm is intended to be complemented by error correction codes. With DNA-Crypt a two-bit message $\mathbf{m} = [m_1, m_2]$ with $m_i \in \{0, 1\}$ is encoded in a single base using $y = f(\mathbf{m}) \in \mathcal{X}$, where the function $f(\cdot)$ is defined by a lookup table such as the one shown in Table I. The cDNA version of DNA-Crypt will be discussed in Section II-B2.

¹Note that this type of frame reading loss would happen for any method under insertion or deletion mutations.

\mathbf{m}	00	01	10	11
$f(\mathbf{m})$	T	C	A	G

Table I: Trivial binary to base encoding

5) *Modified DNA-Crypt*: DNA-Crypt has been used in conjunction with the error correction code Hamming(8,4), for which $r_{\text{ecc}} = 1/2$. Note that the rate in this case is $R_o = r_{\text{ecc}} \cdot R = 1$ bit/base. As highlighted by Heider and Barnekow [3], DNA-Crypt with Hamming(8,4) can only correct $\frac{2}{3}$ substitution mutations that can occur per codeword. This is because $\frac{1}{3}$ of mutations result in 2 bit flips per base. Hamming Code(8,4) is unable to correct a 2 bit error per codeword, resulting in the possible incorrect decoding of 4 bits.

We propose in this subsection a modified DNA-Crypt algorithm for use with Hamming(8,4) that addresses the problem above, while keeping the same embedding rate. Instead of consecutively embedding bits encoded with Hamming(8,4), we propose to embed groups of two bits from consecutive Hamming codewords in a single base. Assume that we have two consecutive Hamming(8,4) binary codewords $\mathbf{c}^{(1)}$ and $\mathbf{c}^{(2)}$, which are of length 8. With the standard DNA-Crypt procedure, the corresponding ncDNA sequence would be formed by the 8 bases $y_k = f([c_{2k-1}^{(1)}, c_{2k}^{(1)}])$ and $y_{4+k} = f([c_{2k-1}^{(2)}, c_{2k}^{(2)}])$ for $k = 1, 2, 3, 4$. With modified DNA-Crypt we propose that the ncDNA sequence be instead $y_k = f([c_k^{(1)}, c_k^{(2)}])$ for $k = 1, 2, \dots, 8$.

If a substitution mutation occurs 1 in every 8 bases this results in no errors after decoding with modified DNA-Crypt. However under the same conditions for the original DNA-Crypt, on average 1 in every 3 substitution mutations that occur cause an error which results in a two bit error per Hamming codeword, which is uncorrectable.

B. cDNA Data Embedding Algorithms

We describe next a number of relevant cDNA algorithms.

1) *Arithmetic encoding*: It was shown in [7] that when codons are uniformly distributed one can embed in a codon three times the same rate as in ncDNA (in bits/base) minus $H(X') = -\sum_{x' \in \mathcal{X}'} \Pr(x') \log_2 \Pr(x')$, that is, the entropy of the random variable representing the primary structure (if codons were strictly independent). Shannon proved that the minimal average number of bits required to encode a independent realisations of a given random variable is given by its entropy. Since arithmetic source encoding can asymptotically achieve the theoretical minimum $H(X')$, then, in the absence of mutations, a cDNA data embedding method based on arithmetic encoding should also be able to yield an optimal code (that is, with maximum embedding rate). The best rate than can be thus achieved is $R = E[\log_2 |\mathcal{S}_{X'}|]$ bits/codon.

Such a method was proposed by Shimanovsky *et al.* [8], well before the considerations above, afforded by the anal-

ysis in [7], were available. This method firstly translates a binary message $\mathbf{m} = [m_1, \dots, m_l]$ to be embedded in \mathbf{x}^c into a real number $r_{\mathbf{m}}$, where $r_{\mathbf{m}} \in [0, 1)$. The information-carrying vector \mathbf{y}^c is then obtained by sequentially selecting codons synonymous to those in \mathbf{x}^c as follows. The encoding algorithm begins by establishing a right-open interval in the real line $\mathcal{I}_1 = [l_1, u_1)$, initialised to $l_1 = 0$ and $u_1 = 1$. At stage k , interval \mathcal{I}_k is subdivided into nonoverlapping right-open subintervals $\mathcal{I}_k^{(i)}$ of equal length $s_k = (u_k - l_k) / |\mathcal{S}_{x'_k}|$ for $i = 1, \dots, |\mathcal{S}_{x'_k}|$, where $x'_k = \alpha(\mathbf{x}_k)$. These subintervals are labelled in a predetermined order with the codons in $\mathcal{S}_{x'_k}$. Using these subintervals, \mathbf{y}_k is chosen as the codon with index j such that $r_{\mathbf{m}} \in \mathcal{I}_k^{(j)}$. This procedure guarantees that $\alpha(\mathbf{y}_k) = \alpha(\mathbf{x}_k) = x'_k$, as required. For the $k+1$ step, a new right-open interval \mathcal{I}_{k+1} is defined with the narrower limits $l_{k+1} = l_k + (j-1) \times s_k$ and $u_{k+1} = l_k + j \times s_k$, where j is the index chosen in step k . This process is repeated throughout the length of the cDNA sequence.

Decoding proceeds using the same subdivision algorithm as above on the mutated sequence \mathbf{z}^c ; the common digits l_k and u_k of the last interval determine $\hat{r}_{\mathbf{m}}$. A serious issue with this method is its extreme sensitivity to mutations. For instance, if a mutation affects a single base in codon \mathbf{y}_i of \mathbf{y}^c the remainder of the mutated sequence $\mathbf{z}_{i+1:n}^c = [z_{i+1}, \dots, z_n]$ can easily become undecodable, since the intervals \mathcal{I}_k are very sensitive to such errors and decoding step k depends on the correctness of decoding step $k-1$.

2) *DNA-Crypt*: The cDNA version of the DNA-Crypt algorithm is based on replacing the third base of certain appropriate codons with a suitable base. As we know, any codon \mathbf{x} is composed of three bases, $\mathbf{x} = [x_1, x_2, x_3] \in \mathcal{X}^3$. An appropriate codon \mathbf{x} is one for which the two following conditions hold: 1) its multiplicity is such that $|\mathcal{S}_{\alpha(\mathbf{x})}| \geq 4$; and 2) for any $x \in \mathcal{X}$, $\alpha([x_1, x_2, x]) = \alpha(\mathbf{x})$, that is, any modification to its third base results in its translation to the same amino acid. For such appropriate codons, this algorithm uses the embedding function $\mathbf{y} = h(\mathbf{x}, \mathbf{m}) = [x_1, x_2, f(\mathbf{m})]$, the function $f(\cdot)$ is the trivial map previously defined for DNA-Crypt with ncDNA and \mathbf{m} is a two-bit message. Condition 2) above guarantees that $\alpha(\mathbf{y}) = \alpha(\mathbf{x})$.

For appropriate codons we trivially have that 2 bits can be embedded. By inspecting the genetic code, one can see that there are only 7 amino acids that contain 4 appropriate codons each. Therefore if all codons are equally likely, that is, if $\Pr(\mathbf{x}) = 1/|\mathcal{X}|^3$, then the average embedding rate of this method is

$$R = 2 \times \frac{7 \times 4}{|\mathcal{X}|^3} = 0.8750 \text{ bits/codon.} \quad (4)$$

With nonuniform codons, as it happens with real genes, one has to compute the corresponding average.

3) *Binary-Codon Equivalency*: We propose in this subsection a novel cDNA embedding method that we call Binary-Codon Equivalency (BCE). Unlike DNA-Crypt, this

perform poorly when subjected to substitution mutations, as shown in Figures 4 and 2. Furthermore due to the way in which embedding takes place adding appropriate error correction is impossible. Huffman code can perform significantly worse depending upon the make-up of the individual codewords, which are determined by the size of the codebook. This is because in some cases it is possible for codewords to mutate into unrecognisable symbols, which may cause decoder errors. The practical application of Huffman or arithmetic coding should not generally be considered, as under certain conditions mutations can destroy the whole embedded message.

IV. CONCLUSION

It is clear from the results of the simulations that any error correction used must be tailored to a particular encoding scheme. Alternating code performs significantly better with the addition of Hamming code (see Figure 3). This is because the alternating code algorithm provides a direct mapping of 1 bit to 1 base which allows Hamming code to provide appropriate error correction. Both alternating code with Hamming(8,4) and modified DNA-Crypt with Hamming(8,4) have very similar error rates. These are much lower than their versions without error correction. The embedding rate of modified DNA-Crypt with Hamming code is twice that of alternating code with Hamming code. This means that modified DNA-Crypt with Hamming code uses half the amount of DNA to embed the same message when compared to alternating code with Hamming code.

In conclusion, the most relevant DNA data embedding algorithms were implemented and their performance under mutations compared for the first time in a meaningful way. New decoding methods were devised where necessary. We have also proposed two new algorithms, shown to both have high embedding rates while maintaining relatively low probabilities of error. However the results in [7] still point at the possibility of much better algorithms. With the constant advances in biological technology it is only a matter of time before data embedding in DNA becomes practical on a larger scale. Indeed, DNA is well suited to data storage: “*The machine code of the genes is uncannily computerlike.*”²

ACKNOWLEDGMENT

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant Number 09/RFP/CMS2212.

REFERENCES

[1] P. C. Wong, K. Wong, and H. Foote, “Organic data memory using the DNA approach,” *Comms. of the ACM*, vol. 46, no. 1, pp. 95–98, January 2003.

²Richard Dawkins River Out of Eden: A Darwinian View of Life.

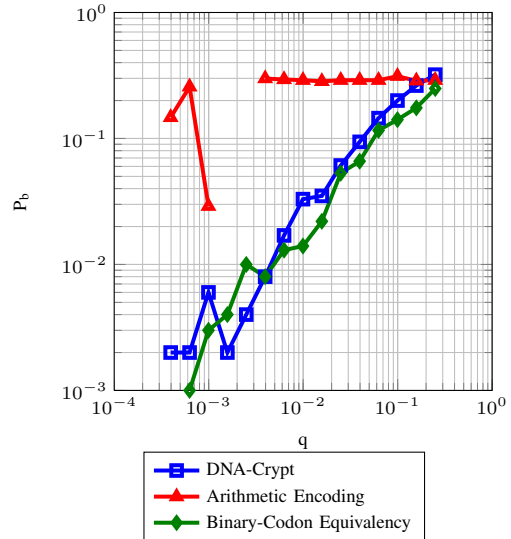


Figure 4: The reason for arithmetic encodings gap in the figure above is that no errors occurred during that simulation for those mutation rates. The average embedding rates are: arithmetic encoding: 0.54259 bits/base; DNA-Crypt: 0.2337 bits/base; Binary-Codon Equivalency: 0.50674 bits/base. The cDNA used was gene RP24-75J17 from a house mouse, Genbank accession No. AC213072.3 .

[2] D. Gibson, G. Benders, C. Andrews-Pfannkoch, E. Denisova, H. Baden-Tillson, J. Zaveri, T. Stockwell, A. Brownley, M. A. D. W. Thomas, C. Merryman, L. Young, V. Noskov, J. Glass, J. Venter, C. Hutchison, and H. Smith, “Complete chemical synthesis, assembly, and cloning of a mycoplasma genitalium genome,” *Science*, vol. 319, pp. 1215–1219, 2008.

[3] D. Heider and A. Barnekow, “DNA-based watermarks using the DNA-Crypt algorithm,” *BMC Bioinformatics*, vol. 8, no. 176, February 2007.

[4] D. C. Jupiter, T. A. Ficht, J. Samuel, Q.-M. Qin, and P. de Figueiredo, “DNA watermarking of infectious agents: Progress and prospects,” *PLoS Pathogens*, vol. 6, June 2010.

[5] N. Yachie, K. Sekiyama, J. Sugahara, Y. Ohashi, and M. Tomita, “Alignment-based approach for durable data storage into living organisms,” *Biotechnol. Prog.*, vol. 23, no. 2, pp. 501–505, April 2007.

[6] G. C. Smith, C. C. Fiddes, J. P. Hawkins, and J. P. Cox, “Some possible codes for encrypting data in DNA,” *Biotech. Lett.*, vol. 25, no. 14, pp. 1125–1130, July 2003.

[7] F. Balado, “On the embedding capacity of DNA strands under substitution, insertion, and deletion mutations,” in *Procs. of the SPIE: Media Forensics and Security II*, vol. 7541, San Jose, USA, 2010.

[8] B. Shimanovsky, J. Feng, and M. Potkonjak, “Hiding data in DNA,” in *Procs. of the 5th Intl. Workshop in Information Hiding*, Noordwijkerhout, The Netherlands, October 2002, pp. 373–386.

[9] R. C. Deonier, S. Tavaré, and M. S. Waterman, *Computational Genome Analysis: An Introduction*. Springer, 2005.