Repetition Coding as an Effective Error Correction Code for Information Encoded in DNA

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

***Abstract*—The goal of DNA data embedding is to enable robust encoding of non-genetic information in DNA. This field straddles the areas of bioinformatics and digital communications, since DNA mutations can be seen as akin to a noisy channel from the point of view of information encoding. In this paper we present two algorithms which, building on a variant of a method proposed by Yachie *et al.*, rely on repetition coding to effectively counteract the impact that mutations have on an embedded message. The algorithms are designed for resynchronising multiple, originally identical, information encoded DNA sequences, embedded within non-coding DNA (ncDNA) sections of a host genome. They use both BLAST and MUSCLE algorithms to accomplish this. Bit error rates at the decoder are established for mutations rates accumulated over a number of generations of the host organism. The empirical results obtained are compared to a theoretical bound for optimal decoding.**

***Keywords*-DNA Data Embedding; Decoding Performance; DNA Watermarking; Sequence Alignment**

I. Introduction

In recent times we have seen the emergence of many storage media for digital data. Over the last decade DNA has been successfully demonstrated by several authors as a viable medium for the storage of non-genetic data [1], [2]. There are strong potential advantages to this use of DNA. If properly encoded, data embedded within an organism's DNA is automatically replicated in new generations, and the lifespan of information thus encoded can reach thousands of years or longer; a scale of time which cannot be rivaled by any modern device.

It is clear that interest in this area is growing. A recent example of which is the IGEM 2010 Bioencryption project[1], which focused on information embedding in DNA because of its high payload. DNA is an extremely compact medium even in comparison to modern memory standards[2].

[1]http://2010.igem.org/Team:Hong_Kong-CUHK

[2]Modern flash memory (NAND): 1.37×10^{14} bits/cm^3; DNA: 1.88×10^{21} bits/cm^3.

Also, genetically engineered bacteria are being created for a wide variety of purposes and it is likely that this trend will continue. To help ensure authorised usage of such bacteria, it has been suggested that watermarks containing patent information be embedded in proprietary DNA sequences [3]. The use of such methods would impose accountability, both ethically and monetarily, on organisations using such technologies. One high profile use of a DNA data embedding method in this context was undertaken at the J. Craig Venter Institute [4]. They successfully encoded authorship information in the DNA of a bacteria artificially synthesised by them.

The most obvious concern with information embedded in DNA is that DNA may undergo mutations, which can alter this information. For the purposes of this paper we will consider insertion, deletion, and substitution mutations. There are natural mechanisms within DNA which, to some extent, enable the correction of mutations. However we should not assume that they are adequate for long-term preservation of non-genetic data embedded within DNA, in particular because this data is not subject to evolutionary pressures —it should be encoded in a manner transparent to biological processes. Digital communications strategies can be used to encode information subject to noise more effectively. In terms of information embedding, substitution mutations are equivalent to bit flips in digital communications, and are a well studied problem. With perfect synchronisation at the decoder, deletions and insertions amount to an erasure channel; in this case error correction codes are not so well developed without perfect synchronisation.

It was first proposed by Yachie *et al.* to use repetition coding as a method of error correction for DNA data embedding [2]. They performed *in vivo* experiments in conjunction with an *in silico* empirical analysis of their method, which successfully demonstrated the feasibility of their proposal. However their empirical analysis relied on knowing exactly the location of the

embedding sites, which may not be the case in reality. It may be that the only way to retrieve this data is via full genome sequencing followed by processing the genome to retrieve the embedded data. Two algorithms designed to do this are proposed in this paper. Furthermore, we evaluate the efficiency of such algorithms both empirically and theoretically.

Implicitly, the mutation model used in Yachie *et al.*'s experiments was the Jukes-Cantor model of molecular evolution, which assumes equal probability of base substitution mutations between all base types. Here we employ instead the more realistic Kimura model of molecular evolution [5], which distinguishes transversion and transition mutations, and which includes the Jukes-Cantor model as a particular case.

Repetition codes are inefficient in comparison to other error correction codes in terms of achieving Shannon capacity for a fixed code length. However nucleotides are abundant in DNA and therefore we can afford a reduction in capacity to ensure robust data retrieval. Finally, repetition coding also has the advantage that the decoding error rate is improved regardless of channel error rate for embedding rates less than channel capacity, and, most importantly, it affords the counteracting of *indels* (i.e., insertions or deletions) by means of realignment methods, as proposed by Yachie et al.[2]

II. Notation and Framework

In the following, upper-case calligraphic letters denote sets, while lower-case bold letters denote vectors. DNA is a essentially a quaternary digital signal composed of four nucleotide base types. DNA nucleotides are given by $\mathcal{X} \triangleq \{\mathrm{A}, \mathrm{C}, \mathrm{T}, \mathrm{G}\}$. Using this set, an entire DNA sequence can be represented as a vector of nucleotides $\mathbf{x} = [x_1, \cdots, x_N]$, $x_i \in \mathcal{X}$.

A. Encoder

A message bit to be encoded is denoted by $m \in \mathcal{M} \triangleq \{0, 1\}$. An encoded nucleotide is given by $y = f(\mathbf{m})$, which is just the translation of a two-bit message $\mathbf{m} \in \mathcal{M}^2$ into a nucleotide using any arbitrary mapping $\{00, 01, 10, 11\} \mapsto \{\mathrm{A}, \mathrm{C}, \mathrm{T}, \mathrm{G}\}$.

For a $2l$-bit message $\overline{\mathbf{m}} = [\mathbf{m}_1, \cdots, \mathbf{m}_l]$ we can produce an l-nucleotide long sequence $\mathbf{y} = [y_1, \cdots, y_l]$, simply by applying the mapping above to pairs of bits. This message-encoding sequence is repeatedly embedded n times at different non-coding sites of a host DNA sequence, similarly[3] to the procedure by Yachie *et al.*[2] Instance i of embedding is represented by $\mathbf{y}^{\underline{i}} = \mathbf{y}$, for $1 \leq i \leq n$. After being subjected to mutations the nucleotide sequence originally corresponding to $\mathbf{y}^{\underline{i}}$ is denoted $\mathbf{z}^{\underline{i}}$. These mutated sequences are used by the decoder to obtain an estimate of $\mathbf{y}$, denoted $\hat{\mathbf{y}}$, from which an estimate of the message can be recovered. As we will see in Section III, ideally this can be done in an optimal way by exploiting a model of the mutations.

In DNA there are two groups of nucleotides called pyrimidines, $\mathcal{Y} \triangleq \{\mathrm{C}, \mathrm{T}\}$ and purines, $\mathcal{R} \triangleq \{\mathrm{A}, \mathrm{G}\}$. Within each of these groups the bases are chemically more similar and substitution mutations between bases within these groups are more likely. A substitution mutation within one of these groups is called a transition. A mutation to a base in an external group is called a transversion. This is is the basis upon which the Kimura mutation model —which is presented in the next section— is built.

The decoding performance for all the algorithms presented here will be measured by means of the probability of bit error (i.e., bit error rate) at the decoder. We will do so both empirically, by means of Monte Carlo simulations, and theoretically. Due to the nature of *indel* mutations, the distance metric used to establish empirical error rates is the edit distance, $d_E(\mathbf{y}, \hat{\mathbf{y}})$. This metric is the minimum number of insertion/deletion operations required to transform one sequence into another. It is necessary to compute probability of error this way, because if the same type of misalignment mutation occurs (i.e., either insertions or deletions) at the same position in the majority of $\mathbf{y}^{\underline{1}}, \mathbf{y}^{\underline{2}}, \ldots, \mathbf{y}^{\underline{n}}$ then the decoded message will have a frame shift in it. This frame shift would cause a much higher bit error rate than appropriate. In other words, if resynchronisation is not completely successful, the empirical error rate would not appropriately represent the true error rate without this strategy.

III. Mutation Model, Optimal Decoder, and Performance Analysis

In order to establish an optimal decoder and theoretically evaluate the error correction ability of the algorithms, it is necessary to use a realistic model of mutations. We will assume that all mutations are mutually independent, that is, that the mutation channel is memoryless. While this premise is sometimes not realistic, it can always be approximated by means of a pseudorandom permutation of the encoded sequences $\mathbf{y}^{\underline{i}}$ shared by encoder and decoder. To model substitution rate frequencies for different base types a cascading Kimura model is used. This model assumes that the information-carrying DNA has been subjected to a cascade of p independent mutation channels corresponding

[3]These authors use instead bit-shifted messages at each repetition instance.

to p generations of the host organism. Mutations in one generation are modelled by the Kimura transition probability matrix $\Pi \triangleq [\Pr(z|y)]$, with $z, y \in \mathcal{X}$, given below:

$$\Pi = \begin{bmatrix} 1-q & \frac{\gamma}{3}q & \frac{\gamma}{3}q & (1-\frac{2\gamma}{3})q \\ \frac{\gamma}{3}q & 1-q & (1-\frac{2\gamma}{3})q & \frac{\gamma}{3}q \\ \frac{\gamma}{3}q & (1-\frac{2\gamma}{3})q & 1-q & \frac{\gamma}{3}q \\ (1-\frac{2\gamma}{3})q & \frac{\gamma}{3}q & \frac{\gamma}{3}q & 1-q \end{bmatrix},$$

where the ordering of columns and rows corresponds to [A, C, T, G]. In the matrix above $q = \Pr(z \neq y|y)$ is the probability of base substitution mutation per generation. The model above can incorporate any given transition/transversion ratio ε by setting $\gamma = 3/(2(\varepsilon+1))$. Following estimates of ε given in [6] for different organisms, γ ranges between between 0.07 and 0.79. The Jukes-Cantor model of molecular evolution can be seen as an instance of the Kimura model when $\gamma = 1$, which according to the range above is less realistic. According to this discussion, the possible values that γ can take when transitions are more likely than transversions are $0 < \gamma \leq 1$.

To obtain the transition matrix after a cascade of p generations (i.e., mutation events), we just multiply Π by itself p times to obtain Π^p. By diagonalising this matrix, it can be readily seen that any entry in Π^p corresponds to one of the following three values:

$$a \triangleq \frac{1}{4}(1+2\mu^p+\lambda^p), \quad b \triangleq \frac{1}{4}(1-\lambda^p), \quad c \triangleq \frac{1}{4}(1-2\mu^p+\lambda^p),$$

where λ and μ are defined as:

$$\lambda \triangleq 1 - \frac{4\gamma}{3}q, \quad \mu \triangleq 1 - 2\left(1-\frac{\gamma}{3}\right)q.$$

In particular, the diagonal entries of Π^p are all equal to a, its skew diagonal entries are all equal to c, while the remaining eight entries are all equal to b. Therefore the substitution mutation rate after p generations is just $q^{(p)} = \Pr(z \neq y|y) = 1 - a = 2b + c$. Of course, $q^{(1)} = q$.

Regarding insertion and deletion mutations, we will assume that they can occur with probability ρ, respectively, per nucleotide and generation. Therefore, after p generations $\rho^{(p)} = 1 - (1-\rho)^p$.

A. Decoder

In order to discuss the decoding strategy, we will assume in the remainder of this section that all instances of $\mathbf{z}^{\underline{i}}$ have been mutually aligned at the decoder. We will actually discuss how to do so in Section IV. Our analysis will be based on the fact that if realignment were perfect, insertions could be ignored and deletions would become erasures. If all messages are equally likely, the optimal decoder in this scenario is the symbolwise maximum likelihood (ML) decoder. For a given two-bit message symbol represented by a nucleotide y, the ML decoder chooses $\hat{y} = \arg\max_{y \in \mathcal{X}} \Pr(\mathbf{z}^{\underline{1:n}}|y)$, where $\mathbf{z}^{\underline{1:n}} \triangleq [z^{\underline{1}}, \cdots, z^{\underline{n}}]$, that is, a vector where $z^{\underline{i}}$ is the element of $\mathbf{z}^{\underline{i}}$ corresponding to the repeated message symbol that we are decoding. Since we are assuming a memoryless mutation channel, $\Pr(\mathbf{z}^{\underline{1:n}}|y) = \Pr(z^{\underline{1}}|y) \cdots \Pr(z^{\underline{n}}|y)$, which just involves transition probabilities from Π^p. If there were k erasures in $\mathbf{z}^{\underline{1:n}}$, then the ML decoder would involve $n-k$ probabilities rather than n, which just amounts to decreasing the repetition factor. It can then be seen that:

$$\Pr(\mathbf{z}^{\underline{1:n}}|y) = \left(\frac{a}{c}\right)^{r_y} \left(\frac{c}{b}\right)^{v_y} b^n, \tag{1}$$

where

$$r_y \triangleq \sum_{i=1}^{n} \delta(y, z^{\underline{i}}), \quad v_y \triangleq \sum_{i=1}^{n} \delta(g(y), g(z^{\underline{i}})), \tag{2}$$

with $\delta(\cdot,\cdot)$ being Kronecker's delta, and $g(\cdot)$ such that $g(x) = 1$ if $x \in \mathcal{Y}$ and $g(x) = 0$ if $x \in \mathcal{R}$. Therefore r_y counts the number of bases in $\mathbf{z}^{\underline{1:n}}$ equal to y, while v_y counts the number of bases in the same vector which belong to the same group as y. Clearly, $v_y \geq r_y$.

In the particular case $\gamma = 1$ (Jukes-Cantor model), $\Pr(\mathbf{z}^{\underline{1:n}}|y) = \left(\frac{a}{b}\right)^{r_y} b^n$, and maximising this conditional probability on y amounts to maximising r_y on y, when $a > b$. Therefore in this case the ML decoder becomes a majority decoder. The advantage of this decoder is that it does not need to know the values of q, γ or p, all of which are required in order to perform ML decoding under the general Kimura model. However we will show in Section V that little is lost by undertaking majority decoding with respect to the true ML decoder if $q^{(p)}$ is small.

B. Theoretical Performance Analysis

The Bhattacharyya bound on the probability of decoded symbol error for 4-ary optimal decoding is given by [7]:

$$P_e \leq \frac{1}{8} \sum_{y \neq y'} \sum_{\mathbf{z}} \sqrt{\Pr(\mathbf{z}|y) \Pr(\mathbf{z}|y')}. \tag{3}$$

With repetition coding, a memoryless channel, and using the transition probabilities of the model, (3) becomes

$$P_e \leq \left(2\sqrt{ab} + 2\sqrt{bc}\right)^n + \frac{1}{2}\left(2b + 2\sqrt{ac}\right)^n. \tag{4}$$

Assuming again that realignment at the decoder is perfect, and denoting by $P_e^{\underline{n}}$ the probability of symbol decoding error under substitution mutations with repetition factor n, we can write the probability of symbol error P_e^{indels} under a mutation channel including both substitutions and indels as $P_e^{\text{indels}} = \sum_{k=0}^{n} \binom{n}{k} (\rho^{(p)})^k (1-\rho^{(p)})^{n-k} P_e^{\underline{n-k}}$. Inserting (4) into this expression, we can see by applying the binomial theorem that

$$\begin{aligned} P_e^{\text{indels}} &\leq \left(\rho^{(p)} + (1-\rho^{(p)})\left(2\sqrt{ab} + 2\sqrt{bc}\right)\right)^n \\ &+ \frac{1}{2}\left(\rho^{(p)} + (1-\rho^{(p)})\left(2b + 2\sqrt{ac}\right)\right)^n . \quad (5) \end{aligned}$$

Conversion of the 4-ary symbol error rate P_e to the bit error rate P_b just requires computing $P_b = \frac{2}{3} P_e$ when the message bits are equally likely.

IV. Realignment Algorithms

The algorithms detailed in this section locate and realign the n sequences $\mathbf{z}^{\underline{i}}$, for $i = 1, \cdots, n$, that lie somewhere within a host genome. Because of mutations, these sequences will be similar but not identical. Once the n alignments have been retrieved, ML decoding is applied as detailed in Section III-A. For this reason, the overall decoder is suboptimal and the theoretical bound (5) will become an approximation rather than a true bound.

To accomplish the realignment task one may consider using a sliding-window alignment method, which would slide along the mutated genome $\mathbf{z}$ comparing window i with all other windows excluding i. This method would store the highest scoring alignment and then lengthen it, in a manner similar to BLAST. However the time complexity of the sliding-window itself is $O(n^2)$. The scoring algorithm needed for each sliding window can be $O(n^3)$ (such as the Needleman-Wunsch algorithm). Due to this complexity issue we have developed two algorithms which use heuristic sequence alignment, and which are described next.

A. Location of Repeated Instances

1) RAlign: The RAlign algorithm proposed in this subsection relies on BLAST [8] as a backbone. It requires that the information be embedded in distinct regions of the genome known to both the encoder and the decoder. One straightforward way to do so is to undertake the information embedding as follows. Firstly the encoding algorithm divides the host genome sequence $\mathbf{x}$ into n same size subsequences, $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$. A suitable site within subsequence $\mathbf{x}_i$ is then used to embed $\mathbf{y}^{\underline{i}}$, replacing the original ncDNA. This overwriting follows the sub-cloning process undertaken in the experiments by Yachie *et al.*

In order to facilitate realignment each $\mathbf{y}^{\underline{i}}$ is prepended and appended with unique 24-nucleotide long sequences. These marker sequences are different between $\mathbf{y}^{\underline{i}}$ and $\mathbf{y}^{\underline{j}}$ for $i \neq j$. This is to ensure that, during realignment, only the information-encoded regions between the markers will be aligned, and not any neighbouring nucleotides. Since BLAST uses default windows sizes of 11 nucleotides, 24 bases is sufficient even with the different types of mutations involved. However for $n > 4$, unique nucleotide sequences are not possible for the bases nearest each instance of $\mathbf{y}$, which may cause the decoder to interpret adjacent nucleotides as part of the information encoded sequence. In order for BLAST to find all possible types of encoded sequences, filtering of low complexity must be turned off.

The decoding algorithm divides the received (mutated) genome $\mathbf{z}$ into n subsequences $\mathbf{z}_1, \cdots, \mathbf{z}_n$ and then performs sequence alignment using BLAST by aligning $\mathbf{z}_i$ against $\{\mathbf{z}_1, \cdots, \mathbf{z}_n\} \setminus \{\mathbf{z}_i\}$, for $i = 1, \cdots, n$, while storing at each such step the highest scoring alignment, which we denote $\tilde{\mathbf{z}}^{\underline{i}}$.

For example, if we wish to embed a message in a host sequence $\mathbf{x}$ using a repetition factor $n = 3$ we would proceed as follows. Firstly we would translate the desired message into an information-carrying nucleotide sequence $\mathbf{y}$. Three copies of this sequence would then be created, that is, $\mathbf{y}^{\underline{i}} = \mathbf{y}$ for $i = 1, 2, 3$, and unique marker sequences would be concatenated to the beginning and end of each $\mathbf{y}^{\underline{i}}$. For instance, $\mathbf{y}^{\underline{1}}$ may have 24 'A' nucleotides prepending it, while $\mathbf{y}^{\underline{2}}$ may have 24 'C' nucleotides prepending it. The sequence $\mathbf{x}$ would then be divided in thirds, and suitable ncDNA sites located for embedding each $\mathbf{y}^{\underline{i}}$ in its corresponding third. Decoding first proceeds by running BLAST, taking $\mathbf{z}_1$ as a query sequence with $\mathbf{z}_2$ and $\mathbf{z}_3$ as database sequences. Then $\mathbf{z}_2$ would be run against $\mathbf{z}_1$ and $\mathbf{z}_3$, followed by $\mathbf{z}_3$ against $\mathbf{z}_1$ and $\mathbf{z}_2$. After each run of BLAST the highest scoring alignment $\tilde{\mathbf{z}}^{\underline{i}}$ is stored. These alignments are then used to reassemble the message using the final stage decoder, which is outlined in Section IV-B.

This method of embedding in distinctly delimited genomic regions may simultaneously serve two purposes, namely, to increase alignment accuracy and to decrease algorithm complexity. One strong advantage this algorithm has is that the only piece of information that the decoder needs to reassemble a message is the repetition factor n.

2) HTAlign: Unlike RAlign, the algorithm proposed in this subsection does not require that the information-

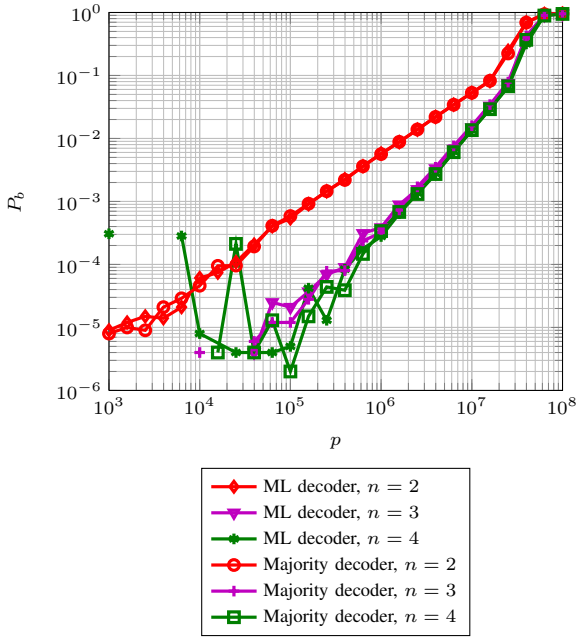

Figure 1: Empirical comparison of ML decoding v.s. majority decoding when $\gamma = 0.1$. RAlign is used before decoding.

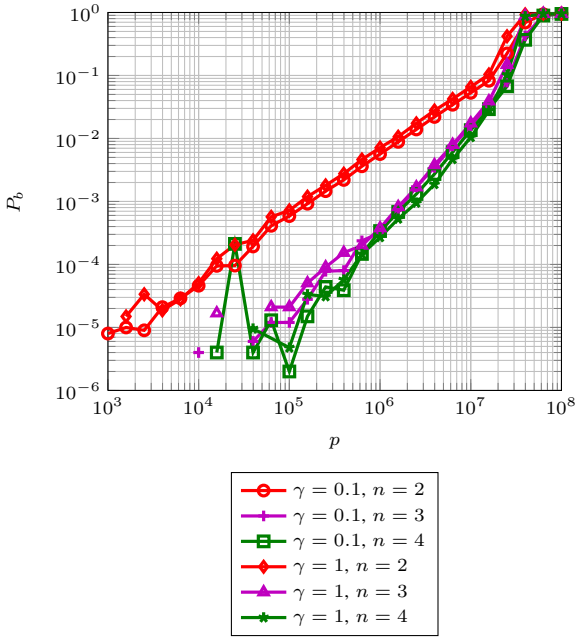

Figure 2: Empirical comparison of majority decoding when $\gamma = 0.1$ and when $\gamma = 1$. RAlign is used before decoding.

carrying sequences $\mathbf{y}^{\underline{i}}$ be confined in prearranged regions of the host genome. To facilitate realignment, pilot sequences known to both encoder and decoder are attached as headers and trailers to the information-carrying sequences. Every header sequence is identical, as is every trailer. Each class of pilot sequence is composed of 40 randomly generated bases. However, unlike RAlign, HTAlign uses BLAST to search for these pilot sequences explicitly. The headers and trailers are then paired with each other, in order of position within the genome. The bases within each of the resulting header-trailer pairs are the $\tilde{\mathbf{z}}^{\underline{i}}$ sequences sought. If the algorithm is successful, it terminates with n such sequences. It is worth mentioning that the use of these pilots may be also convenient as potential primers in DNA sequencing.

Similarly to the example provided for RAlign, if we wish to embed a message with a repetition factor of $n = 3$, we first create three identical information-carrying nucleotide sequences. A randomly generated trailer sequence, identical in all instances, is prepended to $\mathbf{y}^{\underline{1}}$, $\mathbf{y}^{\underline{2}}$ and $\mathbf{y}^{\underline{3}}$. A randomly generated header sequence is also appended, again identical for each instance. The decoder searches for these marker sequences. Once all header and trailer markers are located, they are paired. The nucleotide sequences within each marker pair are the input sequences to the algorithm in Section IV-B.

B. Mutual Alignment of Repeated Instances

The sequences $\tilde{\mathbf{z}}^{\underline{1}} \cdots \tilde{\mathbf{z}}^{\underline{n}}$ found by means of RAlign or HTAlign as described in Section IV-A may contain insertions and/or deletions apart from substitution mutations. Therefore the decoder needs to mutually realign these n sequences by means of a global multiple sequence alignment algorithm, such as MUSCLE [9] or ClustalW. Any of these methods will reveal *indel* sites, after which ML decoding can be performed as described in Section III. Decoding must take into account these frame shift mutations, which may be represented by '-'. If at position i in the encoded sequences there are more than $\frac{n}{2}$ '-' symbols, then it is assumed that an insertion has taken place. Otherwise the '-' symbols are considered to be deletions (i.e., erasures) which modify the repetition factor as discussed in Section III-A.

V. Results

The parameters used in our simulations are as follows. The base substitution rate per generation is $q = 10^{-8}$, while the substitution to *indel* rate is 2.5×10^{-2}, that is, $\rho = 2.5 \times 10^{-10}$, which is an approximation of a ratio given in [10] for bacteria. The message length is 1000 bits, which means $l = 500$ is the nucleotide length of each instance of the repeated information. The host DNA sequence used in these experiments is the

complete genome of an Enterobacteria phage[4], which is 65,955 nucleotides in length. Experimentally we have determined that the MUSCLE algorithm performs with greater accuracy than the Clustal algorithm, and thus is used to obtain our results.

The graphs in this section show probabilities of bit error at the decoder (P_b) versus generations (p). The geometric symbols in the plots represent empirical probabilities, measured by means of Monte Carlo simulations. The repetition factors $n = 2, 3, 4$ were chosen because of their use in Yachie *et al.*'s experiments [2]. Note that it may be impractical to use values of n which are much higher than ones presented here, potentially due to a limited number of suitable ncDNA regions in host organisms with small genomes.

Firstly we empirically compare in Figure 1 the ML decoder against the majority decoder, which is only the true ML decoder when $\gamma = 1$. We can see with this example that the use of majority decoding when $\gamma < 1$ is almost identical to the true ML decoder for all values of n plotted. This supports our use of majority decoding in the remainder. As discussed in Section III-A, the true ML decoder requires a lot of extra information which in practice should be estimated before decoding, leading in turn to decreased performance. Looking at the ML decoder —obtained by maximising the conditional probability (1)— the reason that majority decoding works well is that, if $a > c$ and $c > b$ (as it is typically the case when $\gamma < 1$ and q is small), it is unlikely that if some $y \in \mathcal{X}$ maximises r_y it will not maximise v_y as well, unless the probability of mutation $q^{(p)}$ is very high.

It is clear from the graphs that as n increases, the probability of error decreases. However, the probability of error for $n = 3$ and $n = 4$ is very similar. This is because when n is odd and $n > 1$ adding an extra repeated symbol will not be of much help to the decoder: for odd repetition factors the majority decision can always be unambiguously taken, without the need for breaking ties uniformly at random. An extra symbol will just create many such ties, which are error-prone.

In order to further examine the performance difference of majority decoding for different values of γ, Figure 2 shows an empirical comparison of one such situation. As we have discussed, majority decoding is optimal when $\gamma = 1$ (Jukes-Cantor model). One possible reason for the slightly improved performance when $\gamma < 1$ (which is more clearly visible for $n = 2$) is that decoding decisions are already made before the algorithm reaches the final decoding stage. These decisions include how BLAST and MUSCLE score alignment. Since the Kimura model is more realistic when $\gamma < 1$, it is not surprising that these algorithms can retrieve encoded information with greater accuracy in this case.

We finally observe in Figures 3 and 4 the probability of bit error using both RAlign and HTAlign, compared against our theoretical performance analysis. The empirical results compare well with the Bhattacharyya bounds, which in this case become just approximations and not strictly bounds due to the fact that realignment is not perfect. The bound is tighter for even repetition rates, due to the higher likelihood of ties at the majority decoder in this case. It is likely that the trailing off of the algorithms after 10^7 generations occurs as a result of the algorithms only being capable of partially locate encoded sequences —or not at all— due to the high accumulated mutation rate $q^{(p)}$. Considering the capacity analysis in [11], which shows that the cut-off point is approximately $p \sim 1/(\gamma q)$, we can see that methods better than repetition should be able to yield low probabilities of decoding error up to 10^9 generations.

VI. Conclusions

We have analysed the effect that mutations have on information encoded DNA sequences when repetition coding is used as a means of error correction. The empirical performance, in conjunction with a theoretically established bound attest to the algorithms' accuracy. Furthermore the algorithms perform in reasonable time, with BLAST having a time complexity of $O(l \times (l \times (n-1)))$, MUSCLE a time complexity of $O(n^4 + nl^2)$ and the final stage decoder a time complexity of $O(n)$.

An additional scenario for application of the methods presented here could be the decoding of data using independent colonies of the same organism which shared an information-carrying ancestor. Repetition coding would be naturally provided here by DNA replication. However this could only be accomplished using HTAlign. RAlign requires that encoded information be embedded in prearranged regions, and this may not be necessarily true in the descendants after many generations.

Our findings give further support to the results by Yachie *et al.* showing that repetition is a viable code for embedding information in DNA. The main advantage of repetition is that, while a suboptimal error correction strategy, it greatly simplifies the problem of dealing with *indel* mutations by allowing efficient resynchronisation methods. However, since the gap to capacity is still large [12], the work presented here is a first step towards the development of algorithms which could see data

[4]Genbank accession number: HQ424691

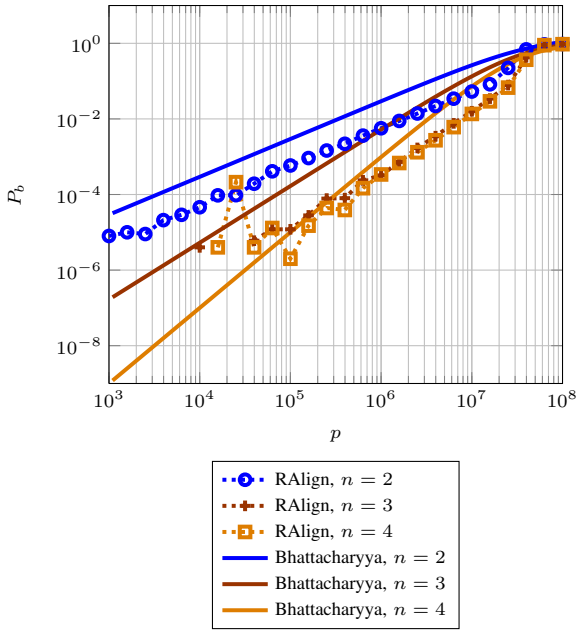

Figure 3: Empirical and theoretical decoding performance using RAlign and majority decoding ($\gamma = 0.1$).

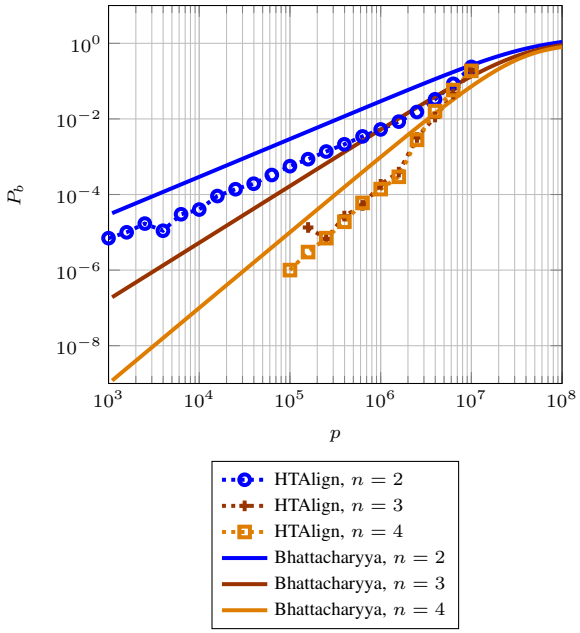

Figure 4: Empirical and theoretical decoding performance using HTAlign and majority decoding ($\gamma = 0.1$).

remaining intact for even longer periods of time. DNA as a mechanism for facilitating this is indeed practical, "*The lifetimes of DNA messages (⋯) are measured in units ranging from millions of years to hundreds of millions of years.*"[5]

Acknowledgment

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

References

[1] C. T. Clelland, V. Risca, and C. Bancroft, "Hiding messages in DNA microdots," *Nature*, vol. 399, no. 6736, pp. 533–534, June 1999.

[2] 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.

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

[5] Richard Dawkins, The Blind Watchmaker.

[4] 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.

[5] M. Kimura, "A simple method for estimating evolutionary rate in a finite population due to mutational production of neutral and nearly neutral base substitution through comparative studies of nucleotide sequences," *J. Molec. Biol.*, vol. 16, pp. 111–120, 1980.

[6] A. Purvis and L. Bromham, "Estimating the transition/transversion ratio from independent pairwise comparisons with an assumed phylogeny," *Journal of Molecular Evolution*, vol. 44, pp. 112–119, 1997.

[7] M. Griott, W.-Y. Weng, and R. Wesel, "A tighter Bhattacharyya bound for decoding error probability," *IEEE Communications Letters*, vol. 11, no. 4, pp. 346–347, Apr 2007.

[8] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *J. Molec. Biol.*, vol. 215, pp. 403–410, 1990.

[9] R. Edgar, "MUSCLE: multiple sequence alignment with high accuracy and high throughput," *Nucleic Acids Research*, vol. 5, no. 32, pp. 1792–1797, 2004.

[10] J. Chen, Y. Wu, H. Yang, J. Bergelson, M. Kreitman, and D. Tian, "Variation in the ratio of nucleotide substitution

and indel rates across genomes in mammals and bacteria," *J. Mol. Biol. Evol.*, vol. 26, no. 7, pp. 1523–1531, 2009.

[11] F. Balado, "Capacity of DNA data embedding under substitution mutations," 2011, arXiv:1101.3457v1.

[12] ——, "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.