SANDIA REPORT

SAND2005-1029
Unlimited Release
Printed March 2005

Deciphering the Genetic Regulatory Code Using an Inverse Error Control Coding Framework

Elebeoba E. May, Anna M. Johnston, Jean-Paul Watson, William M. Brown, and Mark D. Rintoull

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2005-1029
Unlimited Release
Printed March 2005

Deciphering the Genetic Regulatory Code Using an Inverse Error Control Coding Framework

Elebeoba E. May, William M. Brown, Mark D. Rintoul
Computational Biology

Anna M. Johnston, Jean-Paul Watson
Discrete Algorithms and Math

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0310
eemay@sandia.gov

Abstract

We have found that developing a computational framework for reconstructing error control codes for engineered data and ultimately for deciphering genetic regulatory coding sequences is a challenging and uncharted area that will require advances in computational technology for exact solutions. Although exact solutions are desired, computational approaches that yield plausible solutions would be considered sufficient as a proof of concept to the feasibility of reverse engineering error control codes and the possibility of developing a quantitative model for understanding and engineering genetic regulation. Such evidence would help move the idea of reconstructing error control codes for engineered and biological systems from the high risk/high payoff realm into the highly probable/high payoff domain. Additionally this work will impact biological sensor development and the ability to model and ultimately develop defense mechanisms against bioagents that can be engineered to cause catastrophic damage. Understanding how biological organisms are able to communicate their genetic message efficiently in the presence of noise can improve our current communication protocols, a continuing research interest.

Towards this end, project goals include: 1) Develop parameter estimation methods for n for block codes and for n, k, and m for convolutional codes. Use methods to determine error control (EC) code parameters for gene regulatory sequence. 2) Develop an evolutionary computing computational framework for near-optimal solutions to the algebraic code reconstruction problem. Method will be tested on engineered and biological sequences.

Acknowledgement

This work was supported by the Seniors Council Tier 1 Laboratory Directed Research and Development program. We would like to acknowledge and thank John Emerson and Lyndon Pierson for bringing our idea to the attention of the Seniors Council and their continued support of this work.

Contents

Figures

Tables

Summary

We employed information theory, cryptographic analysis, and evolutionary computing methods to achieve the goals of this project. The primary goal was the development of an evolutionary computing framework for determining near optimal solutions for the reconstruction of algebraic error control codes in systematic form. We developed algorithms for the code reconstruction problem. The algorithms were used to study the fitness landscape of the (7, 4) Hamming code. The fitness landscape contains neighborhoods with multiple local maxima. The overall fitness landscape appeared surprisingly regular. Landscape studies helped determine the most viable fitness coefficients for the cost function of our genetic algorithm (GA). The optimal cost function weights the number of zeros in the syndrome matrix of a candidate code significantly higher than the number of non-zero terms in the parity matrix of the code. As the coefficients in the cost function approach optimal values, the (7, 4) code's fitness landscape forms three distinct regions. The majority of codes fall in the 0.5 fitness range followed by the 0.67 range. Very few codes fall in the 0.84 range and only one code (the solution) has a fitness value of 1. Exploration of the fitness landscape also suggested that modifications of candidate solutions in integer space may prove an effective approach for exploring neighborhoods. The reconstruction algorithms were used to develop a GA that uses local search to find the optimal code for a set of codewords.

The secondary objective was to develop parameter estimation methods for key error control code values such as the number of information bits (k), codeword length (n), and memory length for convolutional codes. Expanding the modified entropy approach, we applied the entropy algorithm to codebooks with multiple code configurations to gain insight into codeword length approximation methods. The value of k (number of information bits) in the current algorithm is bounded by log_2(Number of codewords in codebook) for binary codes. Additionally errors in codewords can distort results. For codebooks composed of concatenated codes (a sequence is composed of two codewords produced by two different codes) the coding parameter for the larger code (larger k) masks the k value of the smaller code. Further investigation continues into a more robust and extendable approach for code parameter estimation for a generic sequence set.

In addition to parameter estimation for generic binary codebooks, we explored error control coding characteristics of genetic sequences. More decisive linearity measures for biological sequences were investigated. We developed a biological channel capacity model using a relay channel framework to represent DNA transmission via genetic replication. Using the relay channel framework we found that channel capacity for eukaryotic organisms is less for the entire genome than for the effective genome, suggesting the presence of error control codes of rates less than $\frac{n-1}{n}$ as we originally believed. The relay channel model was also used to perform a cursory study of the relationship between channel capacity and cellular aging and death. The model investigates the correlation between the number of times genomic DNA is replicated in an organism's life time and the capacity of the replication channel. The reduction in capacity over replication time suggests the necessity of significant error control in the long term transmission of DNA.

Deciphering the Genetic Regulatory Code Using an Inverse Error Control Coding Framework

1 Introduction

In the later part of the 1980s the increase in genomic data spurred a renewed interest in the use of information theory in the study of genomics [24, 27, 12]. Information measures based on the Shannon entropy [33] have been used in recognition of DNA patterns, classification of genetic sequences, and other computational studies of genetic processes [24, 22, 2, 29, 28, 3, 26, 21, 9, 32, 34, 23, 16, 30, 31]. Informational analysis of genetic sequences has provided significant insight into parallels between the genetic process and information processing systems used in the field of communication engineering [13, 35, 11, 31].

1.1 The Central Dogma as a Communication System

To determine the algorithm used by living systems to transmit vital genetic information, several researchers have explored the parallel between the flow of genetic information in biological systems and the flow of information in engineering communication systems, re-examining the central dogma of genetics from an information transmission viewpoint [13, 35, 24, 6, 17]. The central premise of genetics is that genes are perpetuated in the form of nucleic acid sequences but function once expressed as proteins [14]. Investigators have developed models that attempt to capture different information theoretic aspects of the genetic system [13, 35, 24, 20, 25].

The inclusion of error control mechanisms is fundamental in the design and implementation of an effective engineering communication system. In a similar manner, one can reason that the inclusion of error control mechanisms is also fundamental to the survival and propagation of biological systems. May et al., discuss the role of error correction in molecular biology as well as various communication system models for molecular biology [19].

Development of coding theoretic frameworks for molecular biology is an ongoing endeavor. Although the existence of redundancy in genetic sequences is accepted and the possibility of that redundancy for error correction and control is being explored and exploited, mathematically determining the encoding algorithm particularly for regulatory regions remains a major research challenge. Devising a method for reconstructing the error control code of a received, noisy, signal is a challenge that if met will provide a way to construct mathematical models of molecular machines (macromolecules such as ribosome, RNA polymerase, and initiation factors) involved in the regulation of genetic processes. To this end, the main focus of this work, which is discussed in the section which follows, is the development of a quantitative framework for reverse engineering error control

codes in engineering systems with the intention of using this framework for discovering and reverse engineering biological error control mechanism. Supporting this primary objective, secondary objectives include further investigation into parameter estimation algorithms for EC sequences and coding theoretic analysis of genetic sequences. Current findings are presented in Section 3.

2 A Quantitative Framework for Reverse Engineering EC Codes

2.1 Linear Block Codes and Generator Matrices

Each codeword, v, in a (n,k) linear block code's codebook can be produced using a generator matrix, G, which encodes the information vector, u, in a deterministic manner [15]. The relationship between u, v, and G is as follows:

$$v = uG \tag{1}$$

where G is k *by* n, u is 1 *by* k, and v is 1 *by* n. The parity-check matrix (also referred to as the dual code of G), H, is a $(n-k)$ *by* n matrix that relates to the generator as follows [15, 4]:

$$GH^T = 0 \tag{2}$$

where H^T is the transpose of the parity-check matrix. As its name suggests, the parity-check matrix is used to check for transmission errors in the received sequence, $r = v+e$. In the absence of errors, $e = 0$, the syndrome vector s (the $n-k$ symbol pattern that results from multiplying the received sequence by the transpose of the parity-check matrix) will be an all zero vector:

$$s = rH^T = (v+e)H^T = vH^T = 0 \tag{3}$$

If $C_{n,k}$ represents the codebook (i.e. contains all codewords v) for a linear (n,k) block code, then based on Equation 3 we can state the following:

$$C_{n,k}H^T = Z \tag{4}$$

where Z is the all zero matrix. Therefore, given a set of codewords produced using a linear block code, it is feasible to determine the dual code, H and ultimately the corresponding generator, G, for the codebook. This is the rational used in constructing a framework for reverse engineering an EC encoded data stream.

To further simplify the process, all linear block codes can be written in systematic form. For systematic (n,k) codes, G and H are of the form

$$G = [I_k; P] \tag{5}$$

$$H = [P^T; I_{n-k}] \tag{6}$$

where P is a k *by* $(n-k)$ matrix and I represents the k *by* k (or $(n-k)$ *by* $(n-k)$) identity matrix [15, 4]. Assuming a systematic code reduces the number of unknowns in the H matrix by $(n-k)^2$. The systematic form also simplifies conversion from H back to G.

In our initial investigation of computational approaches for solving Equation 4 we found that developing an integer program for discovering exact solutions was computationally infeasible. Therefore as a proof of concept to the feasibility of reverse engineering error control codes and the possibility of developing a quantitative model for understanding and engineering genetic regulation we consider evolutionary computing approaches that yield near optimal solutions, specifically genetic algorithms (GA).

2.2 Genetic Algorithms for Reverse Engineering EC Codes [7]

Genetic algorithms are numerical optimization techniques based on a generalized view of the theory of evolution, natural selection, and genetics. Invented in the 1960s by John Holland, GAs have been effectively applied to a wide range of optimization problems of varying size and complexity. Application areas include image processing, three dimensional protein structure predictions, time series analysis, and many other fields. Genetic algorithms perform especially well in problems where the solution space is filled with numerous local optima. The optimal EC code for the translation initiation system resides in such a solution space.

2.2.1 Overview of Genetic Algorithms

An optimization algorithm searches a possibly infinite list of viable solutions, the search space, for the solution(s) that best solves the posed question (i.e. the global optimum). The fitness of a solution is a measure of how successfully the candidate solution solves the problem. To preserve the concept of a search space for multi-dimensional problems, it must be possible to evaluate the fitness of individual solutions and define a measure of the distance between solutions.

Traditional algorithms for locating the optimal solution in a given search space include enumerative searches and direct searches. Enumerative searches estimate the value of the unknown parameter(s) by solving the given problem for a large set of possible solutions. They select the best solution based on minimization of a cost or objective function. Enumerative searches are suitable for problems with a small number of parameters and a rapid algorithm for calculating the objective function. For problems with large search spaces or with computationally intensive objective functions, enumerative searches are not efficient.

Direct searches begin with two possible solutions and based on the value of the objective function at those solution points the next point is selected at a distance δ from the current point. The incremental step size, δ, used to compute the next solution point can be dynamically adjusted. The drawbacks of direct search algorithms include: the algorithm can not be universally applied and the

final solution is dependent on the initial starting point. Direct search algorithms can become trapped in a local optima and fail to locate the optimal solution. In addition, there is a lower confidence associated with the final answer since it is dependent on the algorithm's starting point. Therefore, in complex search spaces with multiple local optima, the direct search algorithm is impractical.

Although simulated annealing algorithms and random searches have been used to find optimal solutions in complex search spaces, random searches guided by concepts that parallel evolution and genetics, have been the most effective of these types of algorithms. Genetic algorithms fall into the random search category.

2.2.2 Components of a Genetic Algorithm

There are four elements which constitute a typical genetic algorithm:

- A population of possible solutions from the problem's solution space. Each possible solution is called an individual. For the linear block code, we define an individual as follows:

```
struct individual {
    /*Define contents of a member of the population;
      stores important info for each individual*/
    int binvec[(N-K)*K];      //binary rep of individual (code)
    double decTag;             //decimal digit rep the individual for tracking
    int gen[K][N];                  //generator matrix
    int h[N-K][N];                  //parity check matrix - dual code of G
    int htrans[N][N-K];         //H transpose
    int p[K][N-K];                   //parity matrix
    int ptrans[N-K][K];          //P transpose
    int nzS;                        //numb zeros in syndrome matrix
    int nzP;                       //numb zeros in parity matrix
    double fitness;            //fitness of individual
    double sfitness;           //scaled fitness of individual
    int tsr;                 //target sampling rate
};
```

  The syndrome matrix is omitted from the individual structure to reduce the memory requirements of the GA.

- A method for evaluating the *fitness* of the individual. *Fitness* is a measure of how well the proposed solution or individual solves the problem being investigated. We use Equation 8 to evaluate the fitness of an individual. Exploration of the (7, 4) Hamming code's fitness landscape, discussed in Section 2.3, provided insight into selecting coefficients for the fitness function.

- An approach for combining the better, or more fit, individuals to form new solution populations with higher average fitness values.
- A mutation method for preserving diversity within the population of individuals.

Genetic algorithms are initialized with a population of usually random possible solutions from the solution search space. The typical size of the initial population ranges anywhere from twenty to one thousand individuals; this number can be smaller or greater depending on the problem. We set the population size to 100, choosing to increase compute cycles and minimize memory requirements. Using three main genetic operators, selection, crossover, and mutation, the initial population "evolves" over a set number of iterations or generations towards convergence to the global optima. The maximum number of generations is set based on the code parameters to ensure only a small fraction of the solution space is evaluated.

Typically, each individual solution is represented as a binary vector called a chromosome. The genetic operators, operate on the chromosome. The binary chromosome is converted to the appropriate representation for the given application and the individual solution is evaluated and assigned a fitness value. Increasing numbers of GAs are using real-valued encodings instead of binary representations for chromosomes. In this work, binary vectors are used to represent P, the parity matrix of individuals in the population.

The following outlines the basic steps for a typical GA:

1. Initialize - Set all probability parameters (including crossover and mutation probability thresholds).
2. Generation=1
3. Create Initial Population - Construct a random population of binary strings (chromosomes).
4. Find Unknowns - Convert the binary chromosomes into the application specific parameters (integers, real numbers, etc.).
5. Assign Fitness - Calculate the fitness of each individual in the population based on some optimization criterion.
6. `For Generation = 2 to MAX_NUMBER_OF_GENERATIONS`
    - *Loop over current population and select pairs of mates*
    - `For New_Individual = 1 to POPULATION_SIZE/2`
        - o Select Parent One
        - o Select Parent Two
        - o Perform Crossover - Produce children (two at a time) by crossing the binary chromosomes of selected parents.

    - Put new individuals into a temporary population
    - `NEXT New_Individual`
  - Mutate - Mutate each individual in the temporary population.
  - Replace - Replace the old (current) population with the new population (contained in temporary population).
  - Find Unknowns - Convert the binary chromosomes (in current population) into the application specific parameters (integers, real numbers, etc.).
  - Assign Fitness - Calculate the fitness of each individual in the current population based on some optimization criterion.
  - `NEXT Generation`
7. END GA

2.2.3 Genetic Operators

There are three main genetic operators used in GAs: selection, crossover, and mutation. Each operator helps move the population towards the optimal solution.

Selection

The selection operator applies pressure to the population similar to natural selection in biological systems. During selection, individuals with high fitness values are selected over low fitness individuals to create the new breeding population. Hence, individuals with high fitness values (good solutions) have a greater than average chance of passing on their information to the next generation. Some of the selection methods used by GAs include:

- Select the top fifty percent of individuals (based on fitness values) for reproduction and discard the rest. This selection technique does not distinguish good individuals from very good individuals. Another drawback is that low fitness individuals are completely annihilated. This reduces the overall genetic diversity of the population.
- Fitness-proportional selection, also referred to as roulette wheel selection, distinguishes between good and very good solutions. In fitness-proportional selection, an individual's probability of selection is proportional to the fitness of the individual. Similar to a roulette wheel, the higher the fitness value of an individual, the larger the arc associated with the individual on a theoretical roulette wheel. The circumference, CIRC, of the wheel is the sum of all fitness values. The spinning of the wheel is simulated by assigning a ball, B, a random number between zero and CIRC. The selection rule is:

$$Sum_k = \sum_{i=0}^{k} f_i$$

  IF $Sum_k > B$ AND $Sum_{k-1} < B$ THEN select individual k
  In the summation equation, f_i represents the fitness value for the ith individual.

- Target sampling rates or *tsr* values can be used to select parents from the current population [5]. Each individual is assigned a fitness-associated target sampling rate. An individual's *tsr* value indicates the number of times they can be selected for mating. For example, an individual with $tsr = 3$ can be selected as a parent three times while a less fit individual with $tsr = 1$ can only be a parent once. Target sampling rates are assigned as follows:

$$tsr(i) = \frac{f_i}{f_{avg}} \tag{7}$$

where f_{avg} is the average fitness for the population. Once all *tsr* values have been assigned, selection proceeds as follows:

1. Select a number, I, between 1 and POPULATION_SIZE.
2. If $tsr(I) > 0$ Then
   - Select individual I
   - Update *tsr*: $tsr(I) = tsr(I) - 1$

Even though the selection of an individual is not fitness-proportional, the ability to reproduce is increased or decreased by an individuals fitness. In current work we use target sampling rates to produce the next generation of solutions.

During each selection iteration, the selection algorithm is applied twice to select a pair of parents to mate using the crossover operator.

Crossover

The crossover operator is a recombination technique that allows individuals in the current population to exchange "genetic" information, similar to the exchange of genetic information by biological organisms during sexual reproduction. In a GA, crossover occurs with probability P_C. Typical values for P_C range from 0.4 to 0.9; we use a crossover rate of 0.7 in this work. There are several crossover methods. The main techniques are described below.

- Single Point Crossover: In single point crossover, the pair of individuals or parents, $P1$ and $P2$, selected using the selection operator are crossed at a single position in their binary chromosomes. Single point crossover proceeds as follows:

1. Generate a random number, p_c, between 0 and 1.
2. If $p_c \leq P_C$ then proceed with crossover (goto next step); else set

$$child1 = P1$$

$$child2 = P2$$

3. Randomly select a position, *POS*, in the chromosome; *POS* will be between 1 and $LENGTH\_CHROMOSOME - 1$.

4. Swap the information to the right of *POS* to produce child1 and child2.

For example, assume the following parents, $P1$ and $P2$, are selected:

$$P1 = 1\ 1\ 0\ 0\ 1\ 0$$

$$P2 = 0\ 1\ 0\ 1\ 0\ 1$$

If the crossover point is randomly selected as,

$$POS = 2$$

Then *child1* is composed of the first two bits of $P1$ and the last four bits of $P2$ while *child2* contains the first two bits of $P2$ and the last four bits of $P1$:

$$child1 = 1\ 1\ 0\ 1\ 0\ 1$$

$$child2 = 0\ 1\ 0\ 0\ 1\ 0$$

In single point crossover, if the parents are identical in the region to the right of the crossover position, the children will be identical to the parents.

- Multipoint Crossover: Multipoint crossover is similar to single point crossover except multipoint crossover allows the selection of multiple crossover points. Given a GA that employs two point crossover, using the same parents from the previous example, if crossover point $POS_1 = 1$ and the second crossover point $POS_2 = 4$ the resulting children would be:

$$child1 = 1\ 1\ 0\ 1\ 1\ 0$$

$$child2 = 0\ 1\ 0\ 0\ 0\ 1$$

- Uniform Crossover: Taking multipoint crossover to its limit, uniform crossover forces bits to be exchanged at every point or locus. This can be disruptive and negatively affect the GA. But parameterized crossover, a form of uniform crossover, applies a probability to each locus to determine whether crossover will occur at that locus[7, 5] The probability of crossover occurring at a locus can range from 0.5 to 0.8. *Note, this is not* P_C.

  As an example, given the above parents, assume the probability of crossover at each locus is represented by the following vector of probabilities:

$$P_{locus} = 0.51\ \ 0.7\ \ 0.22\ \ 0.02\ \ 0.31\ \ 0.1$$

  A crossover mask, the method used by Barnes in [5], can be generated by putting a 0 where the probability is less than the threshold and a 1 where the probability is greater than or equal to the threshold. A 0 indicates a non-crossover locus and a 1 indicates a crossover locus. For a locus crossover threshold of 0.50, the above P_{locus} vector results in the following crossover mask:

$$CrossoverMask = 1\ 1\ 0\ 0\ 0\ 0$$

This particular crossover mask produces the following children given the parents from the single-point crossover example:

$$child1 = 0\ 1\ 0\ 0\ 1\ 0$$

$$child2 = 1\ 1\ 0\ 1\ 0\ 1$$

The crossover operator enables exploration of new regions of the search space.

Mutation

In addition to crossover, mutation helps the GA further explore the search space and possibly frees the GA from local optima solutions. Mutation randomly flips the binary digits in an individuals binary chromosome. Mutation is used sparingly. The probability of mutation (for each binary digit) is determined by P_M which is generally of the order 0.001. The probability of mutation is application dependent. Possible values include:

$$P_M \approx \frac{1}{L_{chromosome}}$$

or

$$P_M \approx \frac{1}{N\sqrt{L_{chromosome}}}$$

where $L_{chromosome}$ is the length of the binary chromosome vector. A mutation rate of 0.005 is used in the current GA implementation, which is less than $\frac{1}{L_{chromosome}}$. Given P_M, the mutation rule is:

- Mutation Rule: For each bit in every individual chromosome, randomly select a number, p, between zero and one. If $p < P_M$ then flip the binary bit (i.e. if current bit is a zero, change it to a one and vice versa); else leave the current binary bit value.

Mutation is the last genetic operator applied to the temporary population prior to fitness assignment. After fitness evaluation, the genetic algorithm can use *elitism* to increase the fitness of the current population.

Elitism

During the run of a genetic algorithm there is a chance that the most fit individual from the previous generation may not be selected for reproduction. It is also possible that all the individuals in the current generation are less fit than the most fit individual (the elite member) from the previous generation. To guarantee that the elite member of the current generation is as fit or more fit than the elite member of the previous generation, a genetic algorithm can employ elitism. Elitism is carried out as follows:

1. If the current generation's elite member is less fit than the previous generation's elite member, proceed to step 2.
2. Randomly select a number, I, between 1 and POPULATION_SIZE.
3. Replace individual I with the elite individual from the previous generation.
4. The elite individual from the previous generation is now the current generation's elite member.

2.2.4 Application to (7, 4) and (16, 11) Hamming Codes

The genetic algorithm was applied to the (7, 4) and (16, 11) Hamming code data set. Table 1 shows the results for execution on a 1GHz PowerPC G4. In Table 1, run time is reported in (h)ours,

Table 1. Result of the GA framework for EC code reconstruction.

Codebook	Run Time	# Gen/Max Generation	Max Fitness
(7, 4) Hamming (T=0)	<1s	9/10	1.00
(7, 4) Hamming (T=1)	1s	10/10	0.8125
(7, 4) Hamming (T=2)	<1s	10/10	0.6875
(7, 4) Hamming (T=3)	<1s	10/10	0.7083
(7, 4) Hamming (T=4)	<1s	10/10	0.7500
(16, 11) Hamming (T=0)	52m 45s	3874/5000	1.00
(16, 11) Hamming (T=1)	1h 7m 15s	5000/5000	0.6807
(16, 11) Hamming (T=2)	1h 7m 34s	5000/5000	0.5539
(16, 11) Hamming (T=3)	1h 9m 27s	5000/5000	0.5333
(16, 11) Hamming (T=4)	1h 8m 35s	5000/5000	0.5342
(16, 11) Hamming (T=5)	1h 8m 37s	5000/5000	0.5394

(m)inutes, (s)econds and the generations column reports the generation where the final optimal solution, *Max Fitness*, occurred out of the maximum possible generations. As T (number of errors in codewords) increased, suboptimal generators were found. For codebook sets with $T > 0$ errors, the GA iterated for the maximum number of generations and the fitness value for the best solution initially decreases with increasing T then increases (after $T = 2$ for the (7, 4) code and after $T = 3$ for the (16, 11) code) with increasing T. A logical explanation for this behavior is that Hamming codes have a minimum Hamming distance, d_{MIN}, value of 3, and errors greater than the code's d_{MIN} value move the codebook out of the current coding sphere into a new coding sphere. The data set is so noisy it has become or is closer to a different EC code, hence the increase in maximum fitness. Studies of the fitness landscape of the (7, 4) Hamming code with $T > 0$ errors are necessary to further understand this phenomenon and validate or refute the explanatory hypothesis.

2.3 Fitness Landscape of a Linear Block Code

Critical to the development of an effective optimization-based framework for EC code reconstruction is the understanding of the fitness landscape of the potential solution space. Exploration of the fitness landscape provides insight into parameters for the fitness function as well as methods for traversing the solution space in an efficient manner.

Using the (7, 4) Hamming code, we explore all $2^{(N-K)*K} = 4,095$ possible linear codes in

systematic form. All possible solutions for P (including $P = Z$, the all zero solution) are interrogated and the fitness returned. The fitness scores H using a cost function of the form:

$$Fitness(H|P) = R_S \frac{\#\ Zeros\ in\ S}{\#\ Elements\ in\ S} + R_P \frac{\#\ Nonzeros\ in\ P}{\#\ Elements\ in\ P} \tag{8}$$

where S represents the syndrome matrix (each row in S corresponds to the syndrome of a codeword in $C_{n,k}$) and $R_S + R_P = 1.0$. Eleven fitness landscapes corresponding to

$$R_S = 0.5, 0.55, 0.6, \ldots, 1.0$$

are explored. Figure 1 shows the coding landscape (decimal representation of P) and corresponding fitness values for $R_S = 0.5, 0.75, 1.0$, respectively. As we move from low to higher R_S values,

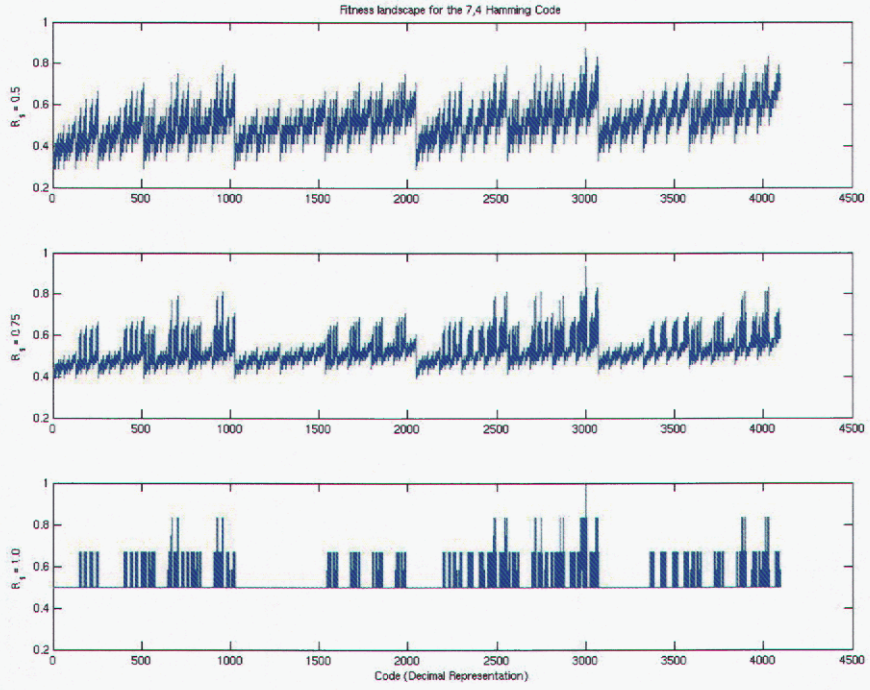

Figure 1. Fitness landscape for the (7, 4) Hamming code for $R_S = 0.5, 0.75, 1.0$.

the coding landscape divides into four discrete fitness regions for $R_S = 1$: 0.5, 0.667, 0.833, and 1.0. Most of the coding solutions fall into the lower fitness categories and only one code, the actual solution, falls into highest fitness strata for the $R_S = 1.0$, $R_P = 0.0$ fitness landscape. For codebook data sets with $T = 0$ errors, the optimal values for the fitness coefficients in Equation 8 are $(R_S = 1.0, R_P = 0.0)$. For error containing variants of the (7, 4) Hamming codes, this may not be the case.

We further analyzed the $(R_S = 1.0, R_P = 0.0)$ fitness landscape of the (7, 4) Hamming code to determine how to efficiently traverse the coding space. Solutions are represented in binary vectors and individual bits are manipulated to generate offspring solutions for GA based optimization (as previously described). The relationship between an individual, P_i

$$P_i = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix}$$

and the corresponding code's generator, G_i is

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & p_1 & p_2 & p_3 \\ 0 & 1 & 0 & 0 & p_4 & p_5 & p_6 \\ 0 & 0 & 1 & 0 & p_7 & p_8 & p_9 \\ 0 & 0 & 0 & 1 & p_{10} & p_{11} & p_{12} \end{bmatrix} \tag{9}$$

The rows of G represent the K basis codewords from which all codewords in the codebook are generated. Changing the bits in P_i changes the basis and the code. Using the fitness landscape data, we evaluated the relationship between Hamming distance of codes and the change in fitness. Figures 2 and 3 show the 3D and 2D distribution of codes with $x = d_{Hamming}(C_i, C_j), \forall i \neq j$ and $y = |d_{Fitness}(C_i, C_j)|, \forall i \neq j$. Our distance-based fitness analysis indicates that $d_{Hamming} = 6$ changes are the most prevalent and result in the largest absolute fitness difference. We also note that the distribution is symmetric, which suggests that with regard to fitness small perturbations in the code are as effective as large perturbations. Additionally, the probability of a random single bit perturbation resulting in a large fitness gain or loss is small. To gain additional insight into the relationship between the number of changes made to a candidate solution and the change in fitness, we analyzed the normalized fitness distribution for each Hamming distance category as shown in Figure 4. In Figure 4 the horizontal axis is change in fitness value and the vertical axis is $Prob(x = \Delta Fitness | d_{Hamming})$. Contrary to our original conclusion, Figure 4 indicates that Hamming distance changes of six or greater are the most probable to impact fitness. Perturbations of three bits or greater have comparable impact on fitness.

We repeated this analysis with respect to the optimum solution, the generator for the (7, 4) Hamming code (see Figures 5, 6, and 7). With respect to the optimal solution, Figures 5 and 6 suggest that the codes with $d_{Hamming} = 3 \ldots 6$ result in the largest fitness gain or loss. Figure 7 further indicates that the majority of two and three bit perturbations of the optimal solution result in the median fitness change of 0.333 and more than approximately 80% of perturbations of $d_{Hamming} \geq 5$ result in the maximum fitness loss of 0.5. This is consistent with the distance properties of the (7, 4) Hamming code, which has a minimum distance value of 3. Hence it is plausible that changes greater than $d_{Hamming} = 3$ result in a significantly different code. Results of the distance-based fitness studies are used to determine neighborhood boundaries for implementing the genetic local search algorithm.

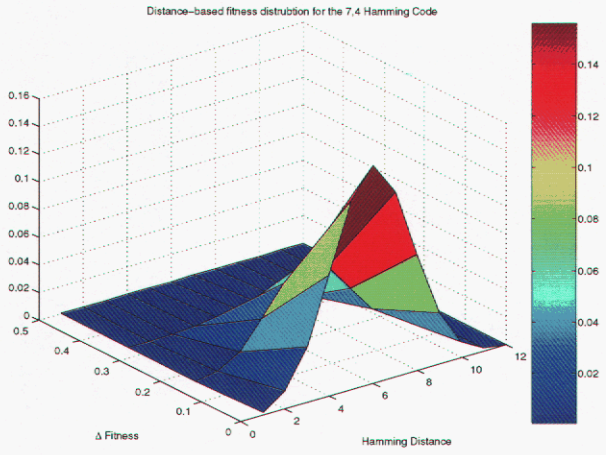

Figure 2. 3D fitness distribution for the (7, 4) Hamming code.

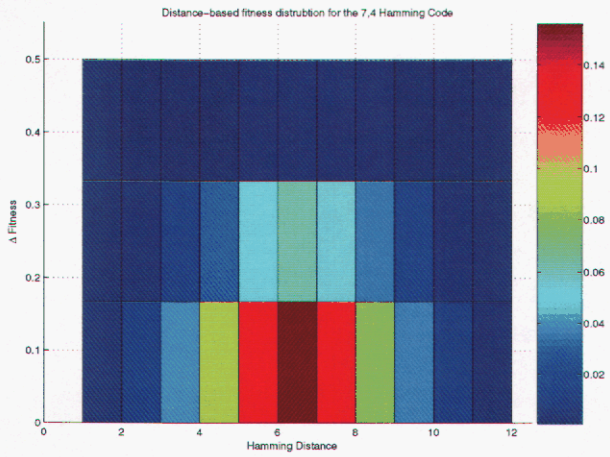

Figure 3. 2D fitness distribution for the (7, 4) Hamming code.

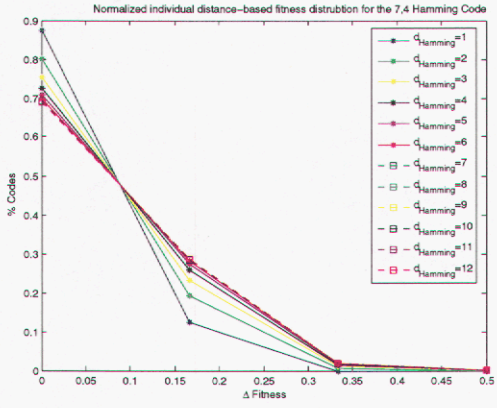

Figure 4. Normalized fitness distribution for the (7, 4) Hamming code.

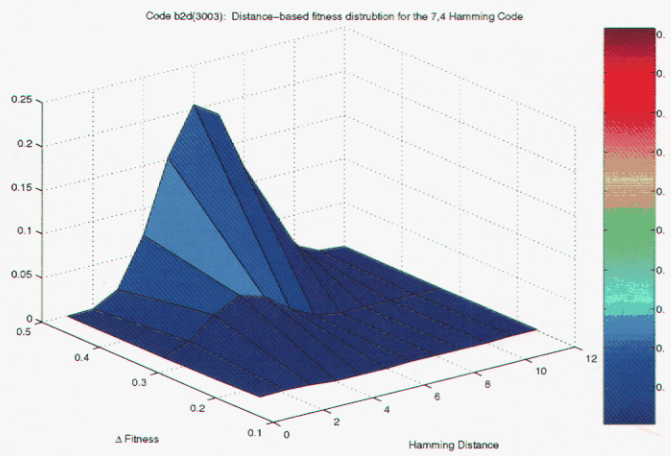

Figure 5. 3D fitness distribution with respect to the optimal solution for the (7, 4) Hamming code.

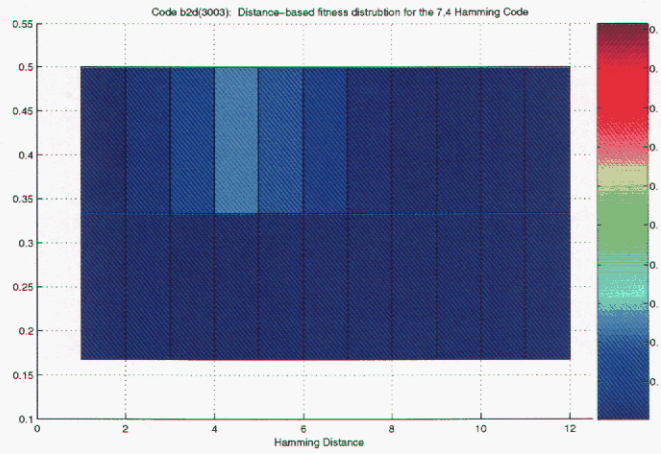

Figure 6. 2D fitness distribution with respect to the optimal solution for the (7, 4) Hamming code.

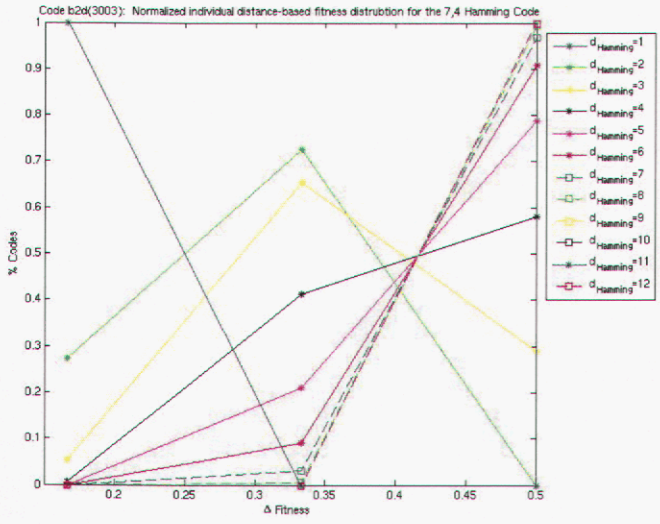

Figure 7. Normalized fitness distribution with respect to the optimal solution for the (7, 4) Hamming code.

2.4 Genetic Local Search

Reconstructing the EC code of a codebook is a combinatorial optimization problem and for error-containing data sets, near optimal solutions may be required. Additionally, analysis of the fitness landscape of the (7, 4) Hamming code revealed surprising regularities and the existence of several local optima in neighborhoods of varying Hamming distances. Given the combinatorial nature and fitness landscape of the code reconstruction problem, incorporation of a local search algorithm is needed. Specifically we implement a genetic local search (GLS) which is derived from the genetic algorithm optimization method.

The two main steps necessary for developing a local search algorithm are problem formulation and neighborhood definition [1]. The problem formulation and fitness function definition developed for the GA optimization framework will be used for the GLS. Building on our initial GA, we define code neighborhoods based on data from the fitness landscape studies previously discussed.

Defining an efficient neighborhood function is problem dependent and critical to the discovery of effective local optima. Honkala and Östergård discuss various neighborhood structures for designing EC codes [1]. Based on a neighborhood structure proposed for finding constant weight EC codes, we define the neighborhood of a solutions as:

Code Neighborhood= All solutions produced by replacing $N-K$ bits of any one of K basis codewords, C, in the current solution with another codeword, C' such that

$$d_{Hamming}(C,C') = D_{MIN}$$

where D_{MIN} is a predetermined Hamming distance.

The K basis codewords are the K rows of the generator matrix, G. We limit changes to the last $N-K$ bits of the codewords to preserve the systematic form of the code. The number of individuals in a neighborhood depends on D_{MIN} as follows:

$$\begin{aligned} Nh_{size} &= K\begin{pmatrix} N-K \\ D_{MIN} \end{pmatrix} \\ &= K\frac{(N-K)!}{D_{MIN}!\ (N-K-D_{MIN})!} \end{aligned} \tag{10}$$

A recursive algorithm was developed to generate all $\begin{pmatrix} N-K \\ D_{MIN} \end{pmatrix}$ Hamming distance combinations for generating neighbors.

The genetic local search algorithm is implemented as follows [1]:

1. Initialize - Construct the initial population of POPULATION_SIZE solutions.
2. Improve - Use local search to replace each solution with POPULATION_SIZE local optima.
3. Recombine - Use crossover and mutation procedures to produce next generation of solutions.
4. Improve - Use local search to replace each solution in the next generation with local optima as applicable. Apply elitism to ensure most fit solution propagates.
5. Evolute - Repeat Steps 3 to 4 until stop criteria reached.

The genetic local search algorithm was implemented and applied to the (7, 4) and (16, 11) Hamming code data set. Table 2 compares the GA and GLS runs on a 1GHz PowerPC G4. The GLS implementation although computationally expensive for larger codes, finds the optimal solution in fewer generations than the GA alone. GLS implementation for the (7, 4) Hamming code shows the potential impact of the local search even when the neighborhood is small. Parallel implementation of the GLS would reduce the computational expense associated with the local search implementation. A parallel execution approach should distribute the local search improvement process among available processor nodes. Efficient methods for minimizing communication cost will be crucial.

Table 2. Comparison of the GLS algorithm to GA framework for EC code reconstruction.

Codebook (D_{MIN})	*GLS* (*RunTime*, *#Generations*)	*GA* (*RunTime*, *#Generations*)
(7, 4) Hamming ($D_{MIN} = 2, Nh_{size} = 12$)	<1s, 2	<1s, 9
(16, 11) Hamming ($D_{MIN} = 3, Nh_{size} = 110$)	13h 2m 23s, 438	52m 45s, 3874
(16, 11) Hamming ($D_{MIN} = 4, Nh_{size} = 55$)	14h 56m 0s, 526	52m 45s, 3874

3 Parameter Estimation

3.1 Estimating (n, k)

Possibilities for n, k are large, hence devising a technique to link properties of the encoded message to characteristics of plausible coding models is vital. We previously developed a modified Shannon entropy method (Equation 11) to determine k, the number of information bits given a (n, k) block code where only n is known [18].

$$H_{avg}^{n,k} = \frac{1}{n - w_k + 1} \sum_{i=1}^{n-w_k+1} \sum_{j=i}^{i+w_k-1} H_j^{n,k} \tag{11}$$

Unlike the case for estimating k, there are no preset bounds on n. The modified Shannon entropy was applied to various EC encoded data sets to determine whether a variation of the Shannon entropy similar to Equation 11 could yield localized or regional asymptotes, indicative of plausible estimates for n.

To determine if and how codeword length affects the entropic profile of the codebook, we applied the modified Shannon entropy to $Codebook_1$, a data set composed of two concatenated (7, 4) Hamming codebooks:

$$Codebook_1 = \left[\; Codebook_{(7,4)Hamming} \;\; | \;\; Codebook_{(7,4)Hamming} \; \right]$$

The concatenated codebook has a length of $n = 14$. The modified entropy method asymptotically approaches the correct $k = 4$ value for $Codebook_1$.

The next data set, $Codebook_2$ is composed of the (7, 4) Hamming codebook followed by a noisy version of the codebook:

$$Codebook_2 = \left[\begin{array}{c} Codebook_{(7,4)Hamming} \\ - - - - - - - - - - - \\ Codebook_{7,4)Hamming,TErrors} \end{array} \right]$$

After testing various instances of the data set, we found that w_k exceeds the $k = 4$ asymptote when $T > 0$ errors. The modified entropy equation is bounded by $\log M$ where M is the number of codewords in the codebook (equivalent to the number of rows in the data set). This result also implies that if the codebook is incomplete, containing less than 2^k codewords for binary codes, the correct value for w_k will not be found. Results for incomplete codebooks lack obvious asymptotic behavior.

To test if we can select multiple values of k in a nested coding sequence, the final data set is composed of a concatenation of the (7, 4) Hamming codebook repeated 128 times and the (16,11) Hamming codebook.

$$Codebook_3 = \left[\begin{array}{c|c} Codebook_{(7,4)Hamming_1} & \\ \hline Codebook_{(7,4)Hamming_2} & \\ \hline \cdots\cdots\cdots & Codebook_{(16,11)Hamming} \\ \hline Codebook_{(7,4)Hamming_{128}} & \end{array}\right]$$

We expected the modified entropy profile for $Codebook_3$ to exhibit multi-asymptotic behavior corresponding to $k = 4$ and $k = 11$ and potentially indicative of plausible n values. We observed a single asymptote corresponding to $k = 11$. The entropic profile of the (16, 11) code overrides that of the (7, 4) code. Based on these results, exploration of two-dimensional modified Shannon entropy measures that incorporate the number of codewords in the data set is a plausible next step.

3.2 Capacity of the Genetic Channel

The capacity of the communication channel is a key system characteristic that governs the type of EC code used in transmission. The genetic communication system presented by May et al., [19] parallels the replication process to an error introducing transmission channel. We revisit the genetic channel capacity question and take into account genome size and the cumulative nature of mutation errors.

3.2.1 Capacity and Replication, Revisited

In previous work, we calculated the channel capacity for various eukaryotic and prokaryotic organisms using the organisms' base mutation rates, μ_b, as reported by Drake et al. [10, 18]. Assuming a discrete memoryless channel (DMC), the capacity of the channel, C, is the maximum reduction in uncertainty of the input X given knowledge of Y [8]:

$$C = \sup_X I(X,Y) \tag{12}$$

where

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{13}$$

The Shannon entropy $H(X)$ and $H(Y|X)$ are defined as:

$$H(X) = -\sum_i p(x_i) log_2 p(x_i) \tag{14}$$

$$H(Y|X) = -\sum_k \sum_j p(x_k, y_j) log_2 p(y_j|x_k) \tag{15}$$

The probability $p(y_j|x_k)$ is the channel error probability. If $p(y|x)$ is specified by the mutation error rate μ_b then $p(y_j|x_k) = \mu_b$, $\forall y \neq x$ and $p(y_j|x_k) = 1 - \mu_b$, $\forall y = x$ (where μ_b is the mutation rate per base per replication cycle). The capacity results for a single base seemed to suggest a near optimal transmission channel and very little reduction in the two bit capacity of the replication channel. This is misleading. Genome replication is not accomplished through a single use of the replication channel. The replication of a genome of size G will require G uses of the replication channel. We can model genome replication as a metachannel with error probability μ_{b_G}, the probability of one or more errors in G uses of the channel. If X_i represents the transmitted bases at channel use i and Y_i represents the corresponding received bases, where $i = 1...G$ and $G = Gsize$, the error probability for the metachannel is derived from μ_b as follows.

$$\begin{aligned} \mu_{b_G} &= \text{Prob}(\text{Y}_i \neq X_i), for\ 1\ or\ more\ (X_i, Y_i)\ pairing \\ &= 1 - \text{Prob}(\text{Y}_i = X_i, \forall i) \\ &= 1 - [(1-\mu_{b_1}) * (1-\mu_{b_2}) * \ldots * (1-\mu_{b_{Gsize}})] \end{aligned} \tag{16}$$

Assuming $\mu_{b_i} = \mu_b$ for all i, we can simplify Equation 16 as follows.

$$\mu_{b_G} = 1 - (1-\mu_b)^G \tag{17}$$

Figures 8, 9, and 10 show the capacity of the meta-replication channel as a function of the log of the organism's genome size for DNA microbes and higher eukaryotes, respectively, using μ_b values from Drake et al. [10] and channel error matrices where the probability of a transition mutation is greater than the probability of a transversion mutation, which is consistent with biological evidence [18]. Prokaryotic organisms have larger channel capacity values than the higher eukaryotes. This suggests that for DNA microbes the coding rate R is closer to $\frac{n-1}{n}$, leaving few bases for EC coding as hypothesized in previous work. In contrast, the channel capacity values for higher eukaryotes implies a distinctly smaller value for R. This implies that the eukaryotic genome has more bases available for EC coding.

3.2.2 Replication Capacity and Cellular Aging

Cell division and mitosis are critical to the growth and survival of multicellular organisms. During mitosis a single cell produces two daughter cells that are identical copies of the parent cell. Vital to the successful production of daughter cells is error-free replication of the DNA contained in the chromosomes of the cell. Although biological replication is highly accurate with minimal

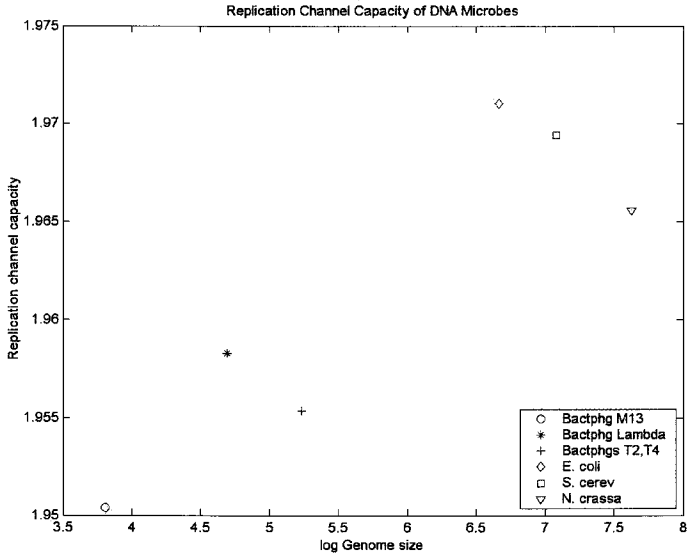

Figure 8. Capacity of prokaryotic replication channels.

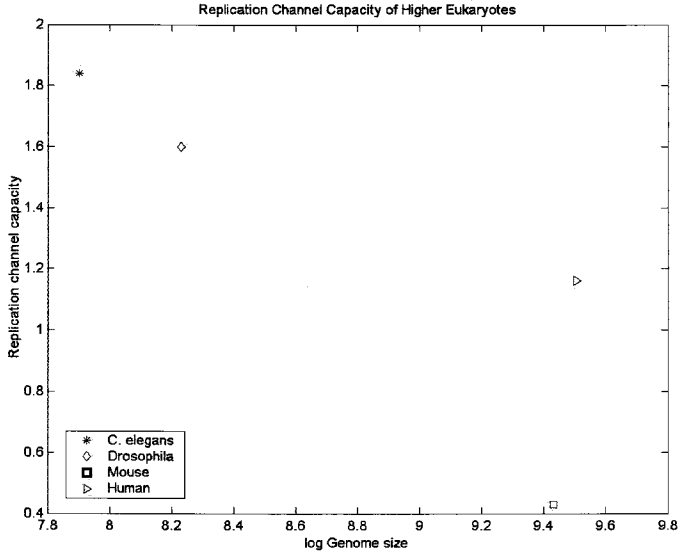

Figure 9. Capacity of eukaryotic replication channels.

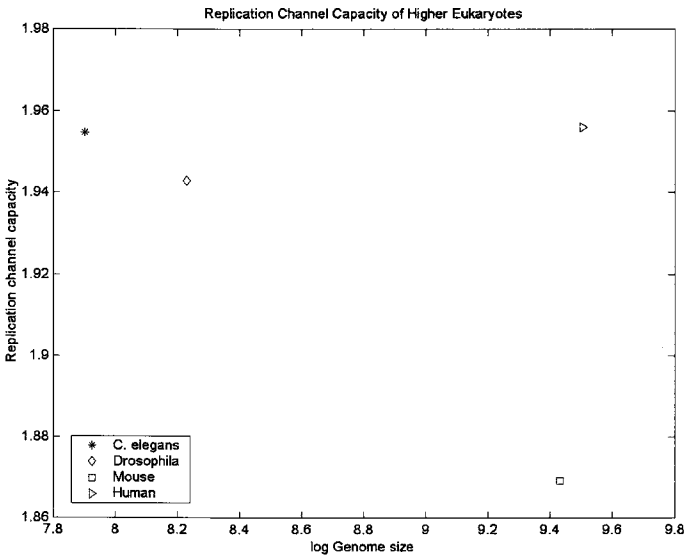

Figure 10. Capacity of eukaryotic replication channels for the effective genome.

error or mutation, errors introduced during the replication process in mitosis propagate to daughter cells. As daughter cells become parent cell and replicate, additional mutations can occur during the generation of grand-daughter cells thereby reducing the fidelity of the original transmitted DNA. To limit the propagation of error-containing DNA, the number of times a chromosome is copied is bounded. Telomeres, the ends of chromosomes, are shortened during each cycle of cell division. Once a cell's chromosomes are shortened to a critical length, that cell can no longer produce daughter cells nor propagate any accumulated mutations. The enzyme telomerase prevents the shortening of telomeres. In normal, adult somatic cells, telomerase is turned off but in some cancerous cells, the telomerase gene is reactivated. We can view aging and related mutation engendered diseases as inevitable communication failures. Extending the meta-replication channel concept, it is evident that for a fixed μ_b, as G increases, the channel error probability also increases. The result is a reduction in channel capacity. Equation 18 is a simple representation of the probability of error for an organisms replication channel after N_{CD} cell divisions.

$$\mu_{bG} \quad = \quad 1 - (1-\mu_b)^{G*N_{CD}} \tag{18}$$

Figure 11 illustrates the reduction in capacity for $N_{CD} = 1 \ldots 75$ cellular generations. A compre-

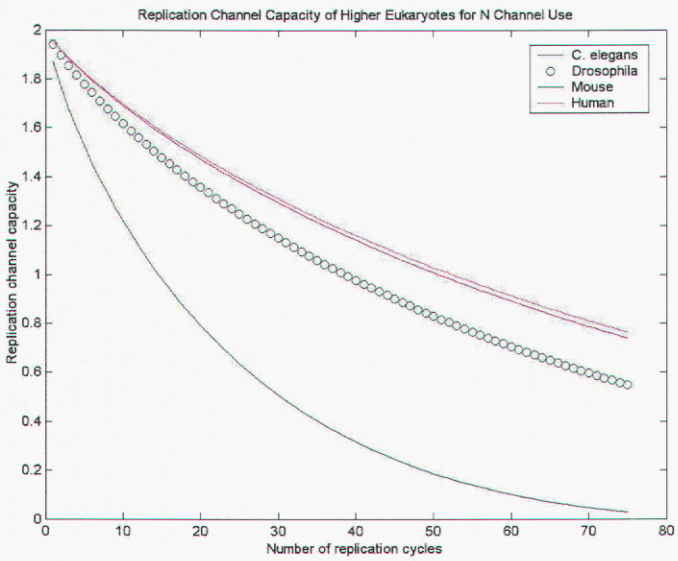

Figure 11. Capacity of eukaryotic replication channel after M cell divisions.

hensive analysis of the replication channel capacity as multiple intput/multiple output relay channel would prove beneficial in forming a communication theoretic understanding of the central dogma of genetics and cell division as related to the emergence of disease. Additionally, correlations be-

tween an organism's capacity for effectively transmitting its genetic information and quantitative measures of virulence (such as minimum infective dose (MID), lethal dose, and morbidity/mortality rates) must be developed. We conducted a preliminary search for mutation databases with sufficient data for constructing replication channel models. Generally, mutation databases can be classified into two types, mutagen induced and loci specific. For a specific gene, mutagen induced databases report the spectrum of changes in the DNA base sequence caused by a particular mutagen under defined conditions. Loci specific databases generally contain known mutations in a gene that result in disease. Further search of mutagen induced databases, such as the Yale Mutation database, may yield the sequence information needed to quantify biological channel characteristics.

3.3 Code detection in RNA sequences

Cryptology, error correcting codes, and data compression are applied to data streams before transmission. Data compression is applied first to minimize the amount of data. Cryptographic algorithms are applied next to secure the data, and error correcting codes are applied last to insure that the data sent is not corrupted with errors during transmission. Given a data stream, such as DNA, the goal here is to determine which, if any, of the techniques is used to encode the data.

The first step in striping off encoding techniques is to determine if error correcting codes were used and if they were, what was the size of the code. Although this seems similar to cryptanalysis on the surface, it is in fact very different. Most cryptanalysis starts with an assumption about what technique was used to encrypt the data. Cryptanalysis tries to find the secret key used in the system. In this case there is no secret key, only an unknown coding technique. As in cryptanalysis, some information about the underlying data must be known if there is any hope of finding the unknown element. The most fundamental piece of information is knowledge of the underlying data type or language. Is it random or is there some inherent redundancy in the language. With DNA/RNA data streams this is a fairly easy piece. There are only 20 amino acids and 64 codons. The character set used in the data stream is the set of 64 codons but the character set of the information transmitted is the set of 20 amino acids. The redundancy stemming from the multiple codons to the smaller number of amino acids is obvious. Less obvious is the redundancy in other parts of the DNA stream in relation to the main information section.

Error correcting codes ensure the correctness of the data by adding redundancy to data streams. The amount of redundancy is an indicator of the size of the error correcting code used. One method for detecting error correcting codes is to determine the underlying redundancy. This can be used as an indicator to the size of the code.

One of the best ways to check for redundancy is with data compression. Applying various compression schemes to the data will give redundancy measures which can then be used to estimate parameters of the error correcting code. Once an estimate for the size of the code is obtained we can use this to assist our search for actual codes used.

File No.	File Names	File size
F1	ham16_11ns.dat	32768
F2	ham74ns.dat	112
F3	CDS.dat	31860
F4	noncds.dat	62000
F5	argc.dat	1104
F6	arog.dat	1152
F7	asna.dat	1092
F8	ilve.dat	1029

The following table describes the compression results, using arithmetic codes, for the data files listed above. The first data column gives the compression ratio using no straight arithmetic codes with no initialization. The following columns give the compression ratios using arithmetic codes initialized with tables generated from the various files. Each entry in a row is also labeled with a number indicating its position in the ordering of the compression ratios, with (1) given to the smallest and (4) given to the largest. Following this ordering is either $\uparrow$ or $\downarrow$. This indicates whether the compression ratio was better than ($\downarrow$ indicating less than) or worse than ($\uparrow$ indicating greater than) the compression using no table.

File No.	No table	F1		F3		F4		F6	
F1	0.1384	0.1358	(1) $\downarrow$	0.1416	(3) $\uparrow$	0.1422	(4) $\uparrow$	0.1411	(2) $\uparrow$
F2	0.5268	0.2232	(1) $\downarrow$	0.2946	(2) $\downarrow$	0.3304	(4) $\downarrow$	0.3125	(3) $\downarrow$
F3	0.2732	0.2734	(4) $\uparrow$	0.2703	(2) $\downarrow$	0.2707	(3) $\downarrow$	0.2702	(1) $\downarrow$
F4	0.2919	0.2910	(4) $\downarrow$	0.2901	(3) $\downarrow$	0.2899	(2) $\downarrow$	0.2898	(1) $\downarrow$
F5	0.3614	0.3524	(4) $\downarrow$	0.2772	(1) $\downarrow$	0.2854	(3) $\downarrow$	0.2826	(2) $\downarrow$
F6	0.3594	0.3516	(4) $\downarrow$	0.2795	(1) $\downarrow$	0.2856	(3) $\downarrow$	0.2830	(2) $\downarrow$
F7	0.3636	0.3599	(4) $\downarrow$	0.2793	(1) $\downarrow$	0.2866	(3) $\downarrow$	0.2848	(2) $\downarrow$
F8	0.3673	0.3547	(4) $\downarrow$	0.2770	(1) $\downarrow$	0.2867	(3) $\downarrow$	0.2847	(2) $\downarrow$

The files $F1$ and $F2$ were formed off a straight error compression data code, while files $F3 - F8$ were from various parts of RNA sequences.

The best compression ratio for $F3$ occured when the table from $F6$ was used. Likewise, the best ratio for $F6$ occured when the table from $F3$ was used. This indicates that the underlying language was closest between these two files. The best compression ratio for $F5 - F8$ occured when the table from $F3$ was used. This probably occured because the underlying language was similar and it was a good deal larger that the other files and therefore a better estimate for the statistics on the language could be formed in the table. Meanwhile, the compression ratios using $F1$ were the lowest of all the tables examined for compressing

These results showed that the underlying language of the straight hamming weight codes ($F1$ and $F2$) did not compare favorably to the underlying language of the RNA sequences. All of the RNA sequence files compressed well ($> .5$) and except for the CDS file compressed using the `ham16_11ns` file, the compression ratios improved using any of the other compression tables. This

high level of redundancy and the relationships between the files could be an indicator of hidden error correction codes

4 Conclusion

This work has laid the foundations for future work on methods for reverse engineering EC codes and research in coding theoretic analysis of genetic systems. We devised a genetic algorithm based computational framework for EC code reconstruction. Incorporation of a genetic local search algorithm significantly reduced the number of generations required to locate the global optima for sample (7, 4) and (16, 11) codebooks. Additional tests using noisy versions of the codebooks are needed to determine the robustness of our current framework. Analysis of the fitness landscape of the (7, 4) Hamming code provided insight for neighborhood construction as well as insight for further improvements of the current GA, such as possible mutation rates and crossover probabilities.

The current framework assumes that the EC coding parameters n and k are given. In this work we investigated methods to determine n and evaluated the robustness of the modified entropy algorithm for inferring k, developed in earlier work. A computationally feasible approach to determine n remains an ongoing but achievable challenge. Studies of the capacity of the replication channel using the relay channel model correlates an organism's genome size and number of cellular divisions with changes in the replication channel capacity. Results refute our original hypothesis of a coding rate of $R = \frac{N-1}{N}$ for eukaryotic organisms. Further investigation using genetic local search based computational framework and insights gained from information theoretic studies is encouraged.

References

[1] Emile Aarts and Jan K. Lenstra. *Local Search in Combinatorial Optimization.* Princeton University Press, 2003.

[2] Hagai Almagor. Nucleotide distribution and the recognition of coding regions in DNA sequences: an information theory approach. *Journal of Theoretical Biology*, 117:127–136, 1985.

[3] Stephen F. Altschul. Amino Acid substitution matrices from an information theoretic perspective . *Journal of Molecular Biology*, 219:555–565, 1991.

[4] John B. Anderson and Seshadri Mohan. *Source and Channel Coding An Algorithmic Approach.* Kluwer Academic Publishers, Boston, MA, 1991.

[5] Tiffany M. Barnes. Using Genetic Algorithms to Find the Best Generators for Half-Rate Convolutional Coding. North Carolina State University, Raleigh, NC, 1994.

[6] G. Battail. Does information theory explain biological evolution? *Europhysics Letters*, 40(3):343–348, November 1997.

[7] David A. Coley. *An Introduction to Genetic Algorithms for Scientists and Engineers.* World Scientific Publishing Co. Pte. Ltd., Singapore, 1999.

[8] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory* . John Wiley and Sons, Inc., New York, N.Y., 1991.

[9] Francisco M. DeLaVega, Carlos Cerpa, and Gariel Guarneros. A mutual information analysis of tRNA sequence and modification patterns distinctive of species and phylogenetic domain. In *Pacific Symposium on Biocomputing*, pages 710–711, 1996.

[10] John W. Drake, Brian Charlesworth, Deborah Charlesworth, and James F. Crow. Rates of spontaneous mutation. *Genetics*, 148:1667–1686, 1998.

[11] Manfred Eigen. The origin of genetic information: viruses as models . *Gene*, 135:37–47, 1993.

[12] T. B. Fowler. Computation as a thermodynamic process applied to biological systems . *International Journal of Bio-Medical Computing*, 10(6):477–489, 1979.

[13] Lila L. Gatlin. *Information Theory and the Living System.* Columbia University Press, New York, NY, 1972.

[14] Benjamin Lewin. *Genes V.* Oxford University Press, New York, NY, 1995.

[15] Shu Lin and Daniel J. Costello Jr. *Error Control Coding: Fundamentals and Applications.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1983.

[16] David Loewenstern and Peter N. Yianilos. Significantly lower entropy estimates for natural DNA sequences. In *Proceedings of the Data Compression Conference*, 1997.

[17] E. May, M. Vouk, D. Bitzer, and D. Rosnick. Analysis of coding theory based models for initiating protein translation in prokaryotic organisms. *BioSystems*, 2003.

[18] Elebeoba E. May, Anna M. Johnston, William E. Hart, Jean-Paul Watson, Richard J. Pryor, and Mark D. Rintoul. Detection and reconstruction of error control codes for engineered and biological regulatory systems, October 2003. SAND2003-3963.

[19] Elebeoba E. May, Mladen A. Vouk, Donald L. Bitzer, and David I. Rosnick. An error-correcting code framework for genetic sequence analysis. *Journal of the Franklin Institute*, 341:89–109, 2004.

[20] Elebeoba Eni May. *Analysis of Coding Theory Based Models for Initiating Protein Translation in Prokaryotic Organisms*. PhD thesis, North Carolina State University, Raleigh, NC, March 2002.

[21] J. L. Oliver, P. Bernaola-Galvan, J. Guerrero-Garcia, and R. Roman-Roldan. Entropic profiles of DNA sequences through chaos-game-derived images . *Journal of Theoretical Biology*, 160:457–470, 1993.

[22] K. Palaniappan and M. E. Jernigan. Pattern analysis of biological sequences. In *Proceedings of the 1984 IEEE International Conference on Systems, Man, and Cybernetics*, 1984.

[23] Angelo Pavesi, Bettina De Iaco, Maria Ilde Granero, and Alfredo Porati. On the informational content of overlapping genes in prokaryotic and eukaryotic viruses . *Journal of Molecular Evolution*, 44(6):625–631, 1997.

[24] Ramon Roman-Roldan, Pedro Bernaola-Galvan, and Jose L. Oliver. Application of information theory to DNA sequence analysis: a review. *Pattern Recognition*, 29(7):1187–1194, 1996.

[25] G. Rosen and J. Moore. Investigation of coding structure in DNA. In *ICASSP 2003*, 2003.

[26] Peter Salamon and Andrezej K. Konopka. A maximum entropy principle for the distribution of local complexity in naturally occuring nucleotide sequences. *Computers and Chemistry*, 16(2):117–124, 1992.

[27] Rina Sarkar, A. B. Roy, and P. K. Sarkar. Topological information content of genetic molecules – I . *Mathematical Biosciences*, 39:299–312, 1978.

[28] Thomas D. Schneider. Theory of molecular machines. I. Channel capacity of molecular machines. *Journal of Theoretical Biology*, 148:83–123, 1991.

[29] Thomas D. Schneider. Theory of molecular machines. II. Energy dissipation from molecular machines. *Journal of Theoretical Biology*, 148:125–137, 1991.

[30] Thomas D. Schneider. Information content of individual genetic sequences . *Journal of Theoretical Biology*, 189:427–441, 1997.

[31] Thomas D. Schneider. Measuring molecular information. *Journal of Theoretical Biology*, 201:87–92, 1999.

[32] Thomas D. Schneider and David N. Mastronarde. Fast multiple alignment of ungapped DNA sequences using information theory and a relaxation method . *Discrete Applied Mathematics*, 71:259–268, 1996.

[33] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication.* University of Illinois Press, Urbana, IL, 1949.

[34] Bonnie J. Strait and T. Gregory Dewey. The Shannon information entropy of protein sequences . *Biophysical Journal*, 71:148–155, 1996.

[35] Hubert Yockey. *Information Theory and Molecular Biology* . Cambridge University Press, NY, NY, 1992.

DISTRIBUTION:

1 MS 0127
Bob Floran, 12111

1 MS 0310
Robert Leland, 9220

1 MS 0310
William M. Brown, 9212

1 MS 0310
Shawn Martin, 9212

7 MS 0310
Elebeoba May, 9212

1 MS 0310
Mark D. Rintoul, 9212

1 MS 1110
John Aidun, 9235

1 MS 0316
Sudip Dosanjh, 9233

1 MS 1110
Steve Plimpton, 9212

1 MS 0321
William Camp, 9200

1 MS 0318
Jennifer Nelson, 9216

1 MS 0521
Richard Damerow, 2561

1 MS 0736
Dana Powers, 6400

1 MS 0785
Tim McDonald, 6514

1 MS 0806
Lyndon Pierson, 5616

1 MS 0819
Tim Trucano, 9211

1 MS 0841
Carl W. Peterson, 9100

1 MS 0847
Scott Mitchell, 9211

1 MS 0885
Grant Heffelfinger, 08330

1 MS 0958
John Emerson, 14172

1 MS 1109
Richard J. Pryor, 9216

1 MS 1110
William E. Hart, 9215

1 MS 1110
Anna M. Johnston, 9215

1 MS 1110
Cynthia A. Phillips, 9215

1 MS 1110
Jean-Paul Watson, 9215

1 MS 1110
David Womble, 9214

1 MS 1111
Bruce Hendrickson, 9215

1 MS 1110
Suzanne Rountree, 9215

1 MS 1165
Lawerence E. Larsen, 15300

1 MS 1190
Craig Olson, 1600

1 MS 1373
Arian Pregenzer, 5320

1 MS 1413
Paul Dressendorfer, 8331

1 MS 1413
Terry Michalske, 8300

1 MS 1744
Susan Brozik, 0892

1 MS 1744
Pat Dolan, 1425

1 MS 9036
William P. Ballard, 8200

1 MS 9217
Steve Thomas, 8962

1 MS 9951
Jean-Loup Faulon, 08333

1 MS 9951
Len Napolitano, 8100

1 MS 9951
Anup Singh, 8130

1 MS 9951
Rajat Sapra, 8130

1 MS 9018
Central Technical Files, 8945-1

2 MS 0899
Technical Library, 9616