CHAPTER 6

Using Genetic Algorithm and Wisdom of Artificial Crowds to Find Hidden Data in DNA

Marc B. Beck, Ahmed H. Desoky, Eric C. Roucka, Patrick S. McClure and Roman V. Yampolskiy

Recent advances in genetic engineering have allowed the insertion of artificial DNA strands into the living cells of organisms. These advances made it possible to insert information into a DNA sequence for the purpose of data storage, watermarking, or communication of secret messages by using substitution ciphers and other methods. While many algorithms have been developed to hide and insert messages into DNA sequences, there are only few approaches of discovering such messages. The ability to detect, extract, and decode messages from DNA is important for forensic data collection and for data security. One of our goals is the development of a software toolkit that employs a combination of several algorithms to decode a message written in DNA symbols. Genetic Algorithms (GA) have traditionally not been successful in solving substitution ciphers, but we have developed a new approach that uses a GA in combination with the Wisdom of Artificial Crowds post processing Algorithm to

Marc B. Beck, Ahmed H. Desoky, Eric C. Roucka, Patrick S. McClure, Roman V. Yampolskiy
University of Louisville
Louisville/Kentucky, USA
e-mail: marc.beck@louisville.edu

overcome these limitations. Our approach is successful in breaking substitution ciphers that encode messages in DNA. Results show that this approach delivers significantly more accurate results than a dictionary approach or a GA by itself.

6.1 Introduction

Deoxyribonucleic acid (DNA) is a molecule carrying the hereditary information for every living organism. DNA contains four different nucleotides distinguished by the bases adenine (A), cytosine (C), guanine (G), and thymine (T) [35]. DNA has the potential to store a vast amount of data over a significant length of time. Data can be encoded using combinations of those four nucleotides within genomes that can range to several billion bases in length [2].

Genomic sequences contain regions that code for genes that produce proteins consisting of amino acids. The four base code of DNA is translated to the twenty base code of amino acids using the genetic code, which was first discovered by Marshall Nirenberg [24].

A codon refers to a sequence in a gene coding region of three nucleotides that determines which amino acid will be produced next during protein synthesis. A total of 43=64 unique codons are possible, given that there are four nucleotides. With exception of the three STOP codons TAA, TAG, and TGA [8], each codon encodes for one of 20 amino acids. This allows for degeneracy in the genetic code where multiple codon sequences translate into the same amino acid. For the purpose of this manuscript, we will refer to encoding patterns that encode alphanumeric characters of messages as codeons in order to distinguish them from codons and avoid confusion.

One of the most important problems in espionage is how to store the obtained information and transfer it out of the target country undetected. An unsuspicious cover medium is needed, and DNA offers itself as such a cover medium. With the appropriate knowledge and technology, a spy could have the information inserted into the DNA of an organism, and send it out of the country as an unsuspicious biological sample. It is possible to insert not only text, but also images and many other forms of digitizable data into a DNA sequence. DNA could also be used by criminal organizations to hide illegal information. These reasons motivated the undertaking of this project to develop a set of forensic tools that can detect, extract, and decode information that has been hidden inside a DNA sequence.

6.2 DNA as Storage Medium

DNA sequencing is becoming increasingly cheaper, faster, and more efficient [18], and at the same time it has become possible to create artificial DNA sequences and insert them into living organisms [14]. These developments make the use of DNA as a stegomedium for concealing, storing, and transmitting messages more feasible. This means that there will be an increasing need for forensic methods to extract and decode such messages in the near future. Not very many such methods are in existence so far.

Table 6.1: Research on data hiding in DNA (modified from [6]).

Researcher	Year	Coding	Message	Location	Organism
Clelland et al. [10]	1999	Substitution	June 6 invasion: Normandy	Artificial	Human
Brenner et al. [8]	1999	Comma code	Not reported	Bsp120I	E.coli
Shimanovsky et al. [28]	2002	Binary to RNA	01001000100100 0101011001001 1010011011101	Theoretical	Theoretical
Wong et al. [36]	2003	Substitution	Not reported	Not reported	Deinococcus radiodurans
Arita and Ohashi [3]	2004	Arita	"AO2KEIO1-F"	ftsZ gene	B. subtilis RIK8
Tanaka et al. [33]	2005	Substitution	"MESSAGE"	Artificial sequence	Artificial DNA strand
Yachie et al. [37]	2007	Keyboard scan	"E=mc^2 1905!"	metB and proB	B.subtilis BEST2136
Heider and Barnekow [19]	2007	DNA-Crypt	"TB"	Vam7 sequence	Saccharomyces cerevisiae CG783
Jiao and Gouette [22]	2009	ASCII 8 bit binary	"CODING"	tatAD gene	B. subtilis
Ailenberg and Rotstein [1]	2009	Improved Huffman	Text: Lyrics "Mary had a little lamb"	SacI/KpnI	PBluescript based plasmid
Ailenberg and Rotstein [1]	2009	Improved Huffman	Music: Tune "Mary had a little lamb"	SacI/KpnI	PBluescript based plasmid
Ailenberg and Rotstein [1]	2009	Improved Huffman	Image: lamb	SacI/KpnI	PBluescript based plasmid
Gibson et al. [14]	2010	Substitution	Multiple messages	Not reported	Artificial bacterium
Mousa et al. [26]	2011	Contrast mapping	Random numbers	RSNn256728	Random Sequence of Nucleotides
Church et al. [9]	2012	Binary to DNA	The book "Regenesis" (53,000 words)	Artificial	Theoretical
Goldman et al. [15]	2013	Goldman	Various files totaling 757,051 bytes	Artificial	Theoretical
Bachelet [4]	2014	Binary	Mona Lisa	Not reported	Mouse
Bachelet [4]	2014	Binary	Entire content of Wikipedia	Not reported	Apple

DNA is being investigated by a number of independent researchers as an ultra-compact medium for long-term data storage [9, 15, 36] (Table 6.1) and as a stegomedium for hiding messages [22, 30]. Instead of representing a message as a sequence of ones and zeroes, it is expressed in DNA code as a series of As, Cs, Gs, and Ts. Researchers have developed various algorithms for encoding a message in DNA code and either disguising it as a novel DNA sequence or inserting it into an existing one. It has been proven possible to insert such artificial DNA components containing encoded information into the genomes of living organisms [3, 8, 10, 14, 19, 21, 22, 37].

The possibility to use the DNA of living organisms as a data storage medium was demonstrated by Yachie et al. [37] in 2007, and Researchers at the J. Craig Venter Institute (JCVI) created the first cell with a synthetic genome, which was capable of reproduction [14]. They managed to insert four watermarks into their artificial genome using a substitution cipher coding scheme. Using DNA as storage medium has many advantages, such as long life, redundancy, and high density as stated by Bancroft et al. [5].

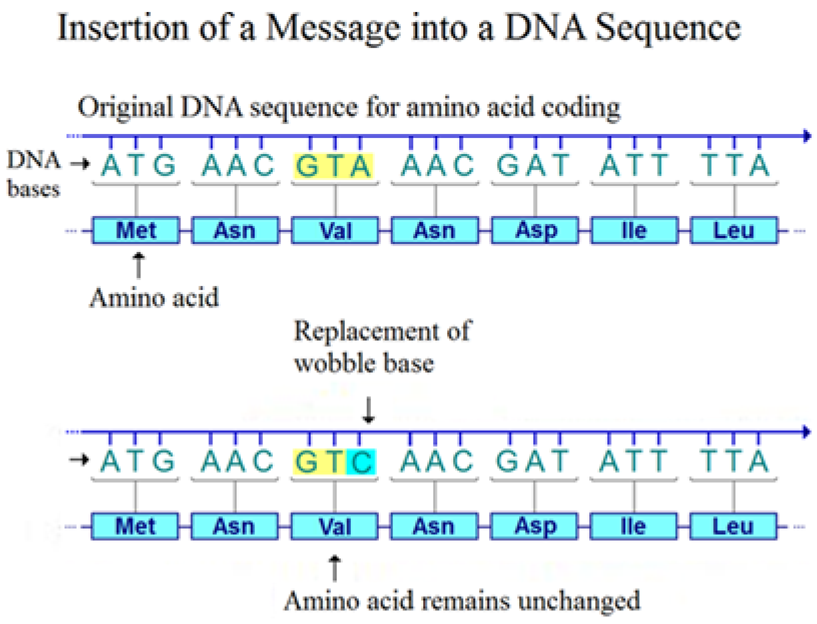
Insertion of a Message into a DNA Sequence
Original DNA sequence for amino acid coding
DNA bases
ATG AAC GTA AAC GAT ATT TTA
Met Asn Val Asn Asp Ile Leu
Amino acid
Replacement of wobble base
ATG AAC GTC AAC GAT ATT TTA
Met Asn Val Asn Asp Ile Leu
Amino acid remains unchanged

Figure 6.1: Message insertion.

6.3 Hiding/Finding Data in DNA

Since noncoding genomic regions might be involved in yet unknown regulations, it was suggested by Arita et al. [3] to insert the message into the coding region of genes. This can be accomplished by taking advantage of the redundancy of the genetic code. In many cases these redundant, or synonymous, codons differ in their third position, called the wobble base [21]. Therefore synonymous codons allow for the potential of message encoding without interfering with the encoded amino acid at the nucleotide level, as illustrated in Fig.(6.1).

6.3.1 Coding Schemes

Several existing coding schemes for encoding messages have been compared and described in great detail by Beck et al. [6]. These coding schemes include two schemes [10, 36] that in effect substitute each letter of the English alphabet, each number from 0-9, and a few special characters with a three letter codeon of DNA symbols. There are two other coding schemes based on the Huffman code [20] and the frequency of letters in the English language [29]. The first encodes only the 26 letters of the English alphabet [30]. The second one is an improved version [1] that encodes all characters on an English computer keyboard. Two other coding schemes are the alternating code [30] and the comma code . The alternating code alternates between purines (Cs and Gs) and pyrimidines (As and Ts) in the codeons that encode the characters. The comma code uses one of the DNA symbols as a comma symbol to separate the codeons. Those coding schemes are less efficient due to their longer codeons.

Furthermore, there are several coding schemes that use various methods to translate a message into binary and use parity bits or other error correction methods before translating the binary into DNA symbols. For Example, the DNA-Crypt coding scheme developed by Heider and Barnekow [19] translates a message into a five bit sequence, where one bit serves as parity bit to keep the respective number of ones and zeroes odd. The other four bits are translated into nucleotides, with two bits per nucleotide. This coding scheme employs the 8/4 Hamming-code and the WDH-code for error correction. The ASCII based coding scheme described by Jiao and Gouette [21] converts each character in the message into its ASCII representation then converts the ASCII code from decimal into binary. Finally, the binary is converted to DNA by replacing 00 with A, 01 with C, 10 with G, and 11 with T.

6.3.2 Solving Substitution Ciphers

In many cases, messages are encoded in DNA using substitution ciphers. For example, the letter 'a' is substituted by the sequence 'AAA', the letter 'b' by 'AAC', and so on. Several methods have been developed for breaking substitution ciphers. These include probabilistic labeling [27], dictionary search [23], combinatorial optimization algorithms [13], maximum likelihood estimator [16], Particle Swarm Optimization (PSO) and Ant Colony Optimization (ASO) [34], as well as various other algorithms [11, 17].

Since there are four nucleotides, a simple substitution cipher coding scheme with a codeon length of three, which can encode 64 characters, can be generated in 64! possible ways. And that is only if the same 64 characters are being used. For example, one coding scheme can start with A=AAA, B=AAC, C=AAG, while another one could be A=AGT, B=CCG, C=CTG, Brute force guessing which of the many permutations has been used to encode the message would take an enormous amount of time and would therefore not be feasible.

6.3.3 Dictionary Approach

Our DNA steganalysis software has two algorithms for solving simple substitution ciphers. In those substitution ciphers, each letter of the English alphabet, numbers from 0-9, and several special characters such as spaces, commas, and periods are each substituted by a combination of three DNA bases. While normal programs for attacking substitution ciphers search over the space of 26! possible keys, our program has a search space of 64! possible keys. This program is capable of solving all possible coding schemes based on substitution ciphers with a codon length of 3 and an alphabet of 64 characters, including the ones developed by Clelland [10] and Wong [36], as well as variations thereof.

The first algorithm uses a list of all the characters in the English alphabet and the frequency of their occurrence in a reference corpus. This corpus contains 801,134 words consisting of 4,899,952 characters, including spaces. The frequency of occurrence in this corpus of the 64 most common characters was recorded. The most common character is the space with 16.2%, followed by the letters E, T, and A with 9.7%, 7.2%, and 6.4%, respectively.

The program will split the cipher text into strings of length 3 and determine the frequency of occurrence of each codon in the cipher text. The program will then assign the most frequent letter from our list to the most frequent codon in the cipher text and so generate a lookup table. Using this lookup table, the program translates the cipher text into the plaintext. Of course most of the plaintext is still nonsense. After that, the program will split the plaintext into words, using an empty space as a delimiter. It will then compare the words with a dictionary. The dictionary, which has been created from the aforementioned corpus, consists of several lists of words; each list contains words with a certain number of letters ranging from single letter words ('a' and 'I') to fifteen letter words. If the word matches a word in the dictionary, it will be left alone. If the word differs from a word in the dictionary by a certain number of letters depending on the length of the word, the program will suggest replacing the letters at that particular position by their counterparts in the correct word. For words with a length of four letters or less, the program will only suggest words that differ by one letter. Words that are five or six letters long will be checked for two letter difference, words with seven or eight letters will be checked for three, and so on.

The program will keep track of which letter is suggested to be replaced by which other letter and how many times. It will then switch the codons of the letter pair that has been suggested for replacement the most often, translate the cipher text into the plaintext with the updated lookup table and repeat checking the dictionary. With each iteration the number of correct words increases. The program terminates when the stop condition selected by the user is met. The possible stop conditions include the following options:

- after a certain percentage of words is decoded correctly.
- after a certain number of iterations have been run.
- after a certain number of words.
- after user clicks the Stop button.

The Stop button is the default setting to prevent the program from running in an infinite loop and can be clicked by the user at any time to terminate the decryption program.

The algorithm follows the following rules:

- An incorrect letter cannot be replaced if it occurs at a different position in the same word and is correct at the second location.
- A letter cannot be replaced if the suggested replacement letter occurs at a different position in the same word and belongs there.
- A replacement word is to be discarded if another replacement is suggested that requires switching fewer letters.
- A word that has a punctuation character at the end will be checked without the punctuation character.

6.3.4 Genetic Algorithm and Wisdom of Artificial Crowds

In order to increase the accuracy with which shorter messages can be decoded we began to search for alternative methods to the dictionary approach. One of the proposed alternatives is to use a Genetic Algorithm (GA). A GA is a heuristic that is commonly used in artificial intelligence to find useful solutions to search and optimization problems. GAs are a subcategory of evolutionary algorithms which mimic natural evolution using concepts such as inheritance, mutation, selection, and crossover. The genetic algorithm contains a population of strings, referred to as chromosomes, which represent candidate solutions. Over several generations (iterations of the algorithm), these evolve from a usually randomly generated population to better solutions. The fitness of every individual in the population is evaluated in each generation. Then multiple individuals are selected based on their fitness, are recombined and occasionally randomly mutated to form a new population, which is then used in the next iteration of the GA. The GA usually terminates when either a satisfactory fitness level has been reached, or after a maximum number of generations has been created.

Spillman et al. [31] developed a GA for the purpose of solving substitution ciphers and reported satisfying results. However, Delman [12], who conducted a study on GAs for solving substitution ciphers, was unable to reproduce their results and after testing several other GAs concluded GAs to be not suitable for this task. In order to address these shortcomings we are combining our GA with the Wisdom of Artificial Crowds (WoAC) [38] post processing algorithm. Yampolskiy et al. [38] developed Wisdom of Artificial Crowds (WoAC) as a post processing algorithm for GA's and Swarm optimization algorithms. It is derived from the Wisdom of Crowds (WoC) algorithm, which is based on the observation that groups are often smarter than the smartest individual in them [32]. For the WoAC algorithm, an nxn occurrence matrix is constructed. This matrix is used to accumulate the number of times each solution appears. Each row number corresponds to a character while each column number corresponds to the symbol it maps to, in this case letters to letters. The occurrence matrix is a symmetric matrix and only the lower triangle is stored in order to save memory. The best key for decoding the message is calculated using the function below:

$$c_{ij} = 1 - I^{-1}_{a_{ij}}\left(b_1, b_2\right) \tag{6.1}$$

where $I^{-1}_{a_{ij}}\left(b_1, b_2\right)$ is the inverse regularized beta function with parameters b_1 and b_2 both taking a value of at least 1 [38].

McClure [**?**] successfully uses a GA in combination with WoAC to attack simple substitution ciphers. We began to modify this GA by increasing the alphabet size from 26 to 64 characters by including numbers and special characters. Also instead of letters substituted for letters we look for strings of DNA symbols substituted for alphanumerical characters.

6.4 Methodology

Both the dictionary approach and the GA with WoAC approach were tested with two different sample messages of different lengths. The first has 202 words, 134 not counting repetition. The second message has 51 words, of which 35 are unique. Both messages have been encoded with the coding scheme developed by Clelland [10].

Our software package was written in Java 7 using Eclipse v.4.4.0. The computer used for this experiment has an Intel Core i7 processor and 10 GB RAM and runs Windows 7 Home Premium 64 bit.

The settings for the GA are as follows: 20 population members, 5000 generations, and a mutation rate of 10%. The results of 10 runs of the GA were entered into the WoAC. Then the results of the WoAC were used to initialize the GA for the next 10 runs, with their results entering into the WoAC again. For the shorter text the GA was run with 1000 generations.

The population size in this approach is 20, with 18 members of the starting population being initialized by creating a random permutation of the English alphabet. The remaining two members were initialized by frequency analysis of the encoded string. In each iteration, the best four members are determined with a fitness score using a dictionary approach and selected as parents. 75% of the time the first and third members are used as parents of the first child and the second and fourth members become parents of the second child. The remaining 25% of the time, the best population member is copied to create a child.

The crossover is performed by choosing a crossover point between 1 and 26 (the number of characters) at random. All elements before the crossover point were copied from one parent, and all elements after the crossover point, if they did not already exist, were copied from the other parent. Elements that have not been filled in so far are copied from the first parent in the same order as they occur there. The mutation rate is 10% and the number of generations is 10,000.

To determine the fitness of each population member, McClure [25] implemented a fitness function that penalizes based on the number of incorrectly spelled words. Furthermore, the misspelling is weighted by the number of letters in the word, and also there is a reward for mapping the letters “e”, ”t”, and “a” to the most common characters in the string, since they are the most commonly used letters in the English language [25].

The keys we produced with ten runs of the GA were fed into the WoAC algorithm as shown in Fig.(6.2). Then we use the key obtained from WoAC as seed value to initialize two out of twenty population members in another run of the GA. The remaining 18 population members are initialized at random.

In order to be able to work with 64 characters instead of 26 we counted the frequency of occurrence of the 64 most common symbols and characters in our sample corpus and adjusted the formula accordingly. Since we take spaces and punctuation into account, our most common character is now the space with 16%, followed by the letters e, t, and a with 9%, 7%, and 6%, respectively. Also, besides rewarding high percentage of occurrence of the most frequent characters, we punish high percentage of occurrence of the 14 least frequent characters. The dictionary used in both approaches contains over 28,000 words.

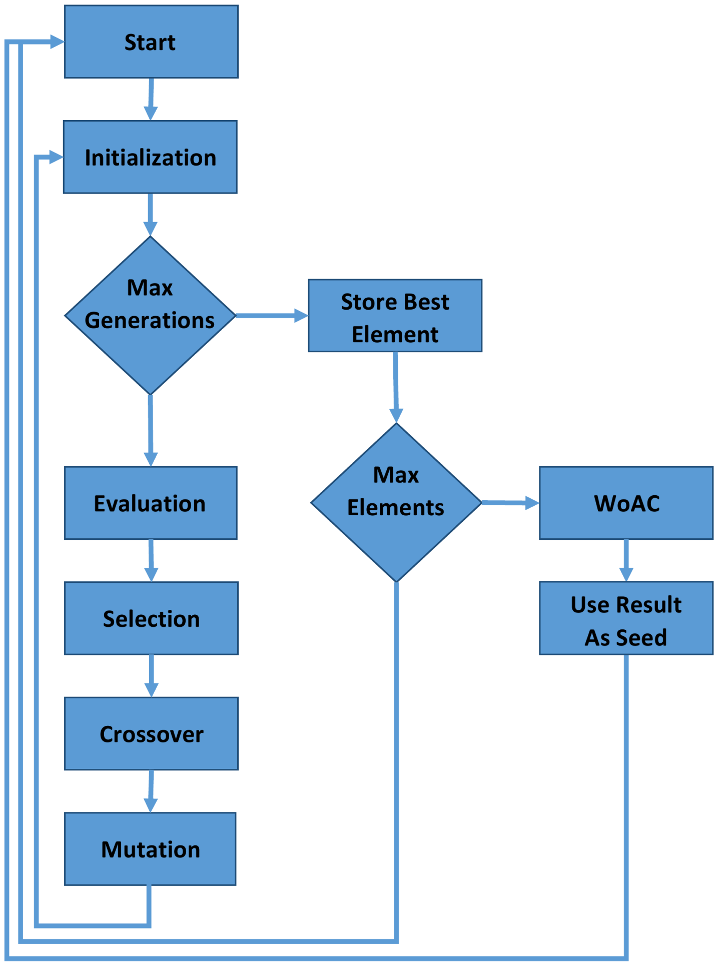

Figure 6.2: Flow chart of GA/WoAC approach.

Table 6.2: Results of decoding messages with the dictionary approach.

Decoded with dictionary approach				
Coding scheme	**Words total**	**Words correct**	**Characters correct**	**Time**
Clelland	202	89%	76%	0.70 sec
Clelland	51	51%	52%	0.06 sec
Comma	202	99%	80%	0.80 sec
Comma	51	64%	42%	0.08sec
Alternating	202	99%	80%	0.90 sec
Alternating	51	58%	50%	0.07 sec

Table 6.3: Results of decoding messages with the GA/WoAC approach.

Decoded with Genetic Algorithm/ Wisdom of Artificial Crowds				
Coding scheme	**Words total**	**Words correct**	**Characters correct**	**Time**
Clelland	202	100%	93%	5hrs 40 min
Clelland	51	89%	74%	7 min 6 sec
Comma	202	100%	96%	5 hrs 33 min
Comma	51	74%	50%	6 min 57 sec
Alternating	202	100%	96%	5hrs 48 min
Alternating	51	84%	53%	7 min 13 sec

6.5 Results

With Clelland's coding scheme the dictionary approach deciphers the text almost correctly, the only errors are that comma and period are switched, and it mistakes the letter J for the number 1. This happens most likely because the frequency of occurrence of the letter J is very similar to the frequency of occurrence of the number 1. The same errors occur with the same message encoded in Wong's coding scheme, but here the program also puts a question mark in the place of an apostrophe. These errors can easily be fixed by adding more rules, for example not to allow numbers in the middle of a word.

The second, much shorter message cannot successfully be deciphered using the dictionary approach. This message has 51 words, 35 of which are unique. Out of those, only 13 words are decoded correctly.

The GA is able to decode the first sample message with 100% accuracy; however each run takes an average of 32 minutes. The shorter message is decoded in 7 minutes with 89% accuracy.

Table 6.4 shows the accuracy for each GA run compared to each other and to the WoAC decoding the 202 word text encoded in Clelland's coding scheme. It also contains the results of the GA runs 11-20 which use the result from the WoAC as seed. The end result at the very bottom of the table is obtained by using the WoAC algorithm on runs 11-20. This table shows the results of the longer of the two texts. Some of the GA runs actually produce worse results individually than the dictionary

Table 6.4: Comparing the separate GA runs to the WoAC results.

Iteration	Words total	Characters Correct
1	99%	90%
2	98%	86%
3	84%	79%
4	10%	34%
5	10%	34%
6	43%	59%
7	73%	69%
8	9%	28%
9	9%	0%
10	96%	79%
WoAC	99%	86%
11	99%	83%
12	99%	83%
13	98%	79%
14	100%	93%
15	100%	93%
16	100%	93%
17	100%	93%
18	99%	90%
19	100%	93%
20	100%	93%
End result	100%	93%

approach. If a word is correct, it means that that particular word can actually be found in the dictionary.

As we can see in run 9, words that are in the dictionary can be decoded even when all the characters are switched. Because of so many "correct" words, the key generated has such a good error score that the algorithm gets stuck in a local maximum. Table 3 also shows how the GA being unable to work itself out of local maxima can result in low success percentages, while at other times the success rates are very high. This makes the GA by itself unreliable; hence Delman's criticism for using GA's to solve substitution ciphers [12]. The table also shows the need for the WoAC to augment the GA in order to produce reliable results. The results of the individual GA runs improve even further when the result of the WoAC is used to seed the GA.

6.6 Further Research

Both approaches still have room for improvement and increased accuracy. The GA could possibly be improved to converge faster by experimenting with different crossover, mutation, and selection algorithms, or a combination thereof. It is also possible to make the GA more dynamic, for instance by changing the mutation rate when it has gotten stuck in a local maximum.

It would be interesting to see how both algorithms perform when attempting to decode messages that have been encrypted before being encoded. More experiments will be run under different parameters in the future, and the knowledge gained from these experiments will be used to further improve our software.

Codons can be in one of three reading frames. We are planning to improve our message detection algorithm to be able to search for messages in all reading frames. Furthermore, insertions and deletions can shift a message mid-stream and an algorithm needs to be developed to take this into account.

Bharadwaj et al. [7] applied for a patent for software that would enable all 256 Extended ASCII characters to be defined in terms of DNA sequences. Therefore, our DNA steganography toolkit could be expanded to be able to deal with alphabets of more than 64 characters.

We have also written a program that creates a dictionary and calculates the frequency of occurrence of characters from a sample corpus, allowing this program to be easily modified by creating a dictionary and a frequency count based on a specific topic.

6.7 Conclusion

This project aims at covering a variety of different approaches for DNA steganography. The GA clearly takes several orders of magnitudes more time, but is able to decode short messages at greater accuracy than the dictionary approach. Both methods complement each other, while the dictionary approach is faster; the GA is more accurate and also performs better at decoding shorter messages. Shorter messages need more generations but take less time. The WoAC algorithm used in combination with multiple runs of the GA provides clearly improved results compared to the individual runs of the GA by themselves. The end result is even better after using the WoAC results as seed for the next 10 GA runs. Both algorithms are currently being improved further and tested with a greater variety of messages, such as messages with a large amount of numbers and special characters and messages that contain names and foreign words. The key is to make this software toolkit as flexible as possible, so it can be adapted in the future to deal with new coding schemes and new approaches to hide messages.

References

[1] M. Ailenberg and O. Rotstein. An improved Huffman coding method for archiving text, images, and music characters in DNA. *Biotechniques*, 47(3):747–754, 2009.

[2] B. Anam, K. Sakib, Md.A. Hossain, and K. Dahal. Review on the advancements of DNA cryptography. In *International Conference on Software, Knowledge, Information Management and Application*, 2010.

[3] M. Arita and Y. Ohashi. Secret signatures inside genomic DNA. *Biotechnology Progress*, 20(5):1605–1607, 2004.

[4] I. Bachelet. Dr. ido bachelet talk on bionic technologies. `http://personalitycafe.com/science-technology/439362-dr-ido-bachelet-talk-bionic-technologies.html`, 2015. Personality Cafe.

[5] C. Bancroft, T. Bowler, B. Bloom, and C.T. Clelland. Long-term storage of information in DNA. *Science*, 293(5536):1763–1765, 2001.

[6] M B. Beck, E.C. Rouchka, and R.V. Yampolskiy. *Digital Forensics and Cyber Crime*, volume 114 of *LNICST*, chapter Finding data in DNA: computer forensic investigation of living organisms, pages 204–219. Springer Berlin Heidelberg, 2013.

[7] L.M. Bharadwaj, A.K. Shukla, A.P. Bhondekar, R. Kumar, and R. P. Bajpai. Method for storing information in DNA, 2005.

[8] S. Brenner, A.O.W. Stretton, and S. Kaplan. Genetic code: the 'nonsense' triplets for chain termination and their suppression. *Nature*, 206:994–998, 1965.

[9] G.M. Church, Y. Gao, and S. Kosuri. Next-generation digital information storage in DNA. *Science*, 337(6102):p. 1628, 2012.

[10] C.T. Clelland, V. Risca, and C. Bancroft. Hiding messages in DNA microdots. *Nature*, 399:533–534, 1999.

[11] F.H.C. Crick and L.E. Orgel. Directed panspermia. *Icarus*, 19(3):341–346, 1973.

[12] B. Delman. Genetic algorithms in cryptography. Master's thesis, Computer Science Department, Rochester Institute of Technology, USA, 2004.

[13] W.S. Forsyth and R. Safavi-Nani. Automated cryptanalysis of substitution ciphers. *Cryptologia*, 17(4):407–418, 1993.

[14] D.G. Gibson, J.I. Glass, C. Lartigue, V.N. Noskov, R.Y. Chuang, M.A. Algire, and et al. Creation of a bacterial cell controlled by a chemically synthesized genome. *Science*, 329(5987):52–56, 2010.

[15] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E.M. LeProust, B. Sipos, and E. Birney. Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. *Nature*, 494:77–80, 2013.

[16] G.W. Hart. To decode short cryptograms. *Communications of the ACM*, 37(9):102–108, 1994.

[17] S. Hasinoff. Solving substitution ciphers. Technical report, University of Toronto, Canada, 2003.

[18] E.C. Hayden. The $1000 genome. *Nature*, 507(7492):294–295, 2014.

[19] D. Heider and A. Barnekow. DNA-based watermarks using the DNA-Crypt algorithm. *BMC Bioinformatics*, 8:176, 2007.

[20] D.A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[21] S.-H. Jiao and R. Goutte. Code for encryption hiding data into genomic DNA of living organisms. In *9th International Conference on Signal Processing (ICSP)*, pages 2166–2169, 2008.

[22] S.-H. Jiao and R. Goutte. Hiding data in DNA of living organisms. *Natural Science*, 1(3):181–184, 2009.

[23] M. Lucks. A constraint satisfaction algorithm for the automated decryption of simple substitution ciphers. In *Advances in Cryptology (CRYPTO)*, pages 132–144, 1988.

[24] R.G. Martin, J.H. Matthaei, O.W. Jones, and M.W. Nirenberg. Ribonucleotide composition of the genetic code. *Biochemical and Biophysical Research Communications*, 6(6):410–414, 1962.

[25] P. McClure. Project 6: Genetic Algorithm with Wisdom of Artificial Crowds for Homophonic Substitution Ciphers. Technical report, University of Louisville, 2012.

[26] H. Mousa, K. Moustafa, W. Abdel-Wahed, and M. Hadhoud. Data hiding based on contrast mapping using DNA medium. *The International Arab Journal of Information Technology*, 8(2):147–154, 2011.

[27] S. Peleg and A. Rosenfeld. Breaking substitution ciphers using a relaxation algorithm. *Communications of the ACM*, 22(11):598–605, 1979.

[28] B. Shimanovsky, J. Feng, and M. Potkonjak. *Information Hiding*, volume 2578 of *LNCS*, chapter Hiding Data in DNA, pages 373–386. Springer Berlin Heidelberg, 2003.

[29] S. Singh. *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots to Quantum Cryptography*. Doubleday, New York - USA, 1999.

[30] G.C. Smith, C.C. Fiddes, J.P. Hawkins, and J.P.L. Cox. Some possible codes for encrypting data in DNA. *Biotechnology Letters*, 25(14):1125–1130, 2003.

[31] R. Spillman, M. Janssen, B. Nelson, and M. Kepner. Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers. *CryptologiaS*, 17(1):31–44, 1993.

[32] J. Surowiecki. *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*. Doubleday, New York - USA, 2004.

[33] K. Tanaka, A. Okamoto, and I. Saito. Public-key system using DNA as a one-way function for key distribution. *Biosystems*, 81(1):25–29, 2005.

[34] M.F. Uddin and A.M. Youssef. An artificial life technique for the cryptanalysis of simple substitution ciphers. In *Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1582–1585, 2006.

[35] J.D. Watson and F.H.C. Crick. Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. *Nature*, 171:737–738, 1953.

[36] P.C. Wong, K.-K. Wong, and H. Foote. Organic data memory using the DNA approach. *Communications of the ACM*, 46(1):95–98, 2003.

[37] N. Yachie, K. Sekiyama, J. Sugahara, Y. Ohashi, and M. Tomita. Alignment-based approach for durable data storage into living organisms. *Biotechnology Progress*, 23(2):501–505, 2007.

[38] R.V. Yampolskiy, L. Ashby, and L. Hassan. Wisdom of artificial crowds – a metaheuristic algorithm for global optimization. *Journal of Intelligent Learning Systems and Applications*, 4(2):98–107, 2011.