# An Investigation into the Zodiac Ciphers using Genetic Algorithms

Final Report for CS39440 Major Project

*Author*: Martin Key (mak18@aber.ac.uk)
*Supervisor*: Dr. Richard Jensen (mkj@aber.ac.uk)

06 May 2014

Version 1.0

This report is submitted as partial fulfilment of a BSc degree in Computer Science (G400)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

## Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the sections on unfair practice in the Students' Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
- I understand and agree to abide by the University's regulations governing these issues.

Signature …………………………………………….. (Martin Key)

Date …………………………………………………

## Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Signature …………………………………………….. (Martin Key)

Date …………………………………………………

## Abstract

The aim of this project was too look into methods in which genetic algorithms could be used in the cryptanalysis of ciphers, specifically those written by the self styled serial killer the 'Zodiac'. There have been a number of prior attempts to use genetic algorithms in cryptanalysis, and even the zodiac ciphers have been attempted, but these mostly focus on simple substitution ciphers. By incorporating the ideas on analysing ciphers from these papers it is concluded that it should be possible to create a  genetic algorithm which would attempt to solve a homophonic cipher; the assumed enciphering technique for the Zodiac 340 cipher. By implementing a genetic algorithm with a number of traditional genetic algorithm operators, it was possible to assess which would be most useful in attempting the zodiac ciphers, by assessing a simpler option a simple substitution cipher. Overall, the algorithm was not useful in the decryption of substitution ciphers or the Zodiac ciphers. Improvement in the average population fitness was seen while attempting the substitution cipher, which suggest this methodology could be useful. However a consistent fall in average population fitness over the investigation of the Zodiac ciphers suggests that a lot of further development in the area would be needed to find a viable solution.

## Contents

## Figures

## Tables

# 1 Introduction

## 1.1. Overview

The murders of David Faraday and Betty Lou Jensen on Friday Dec. 20, 1968[16] began the shocking killing spree of a man naming himself "The Zodiac". Over the next 5 years, there would be another 3 deaths attributed to this mystery man, and a possible connection to 4 other victims[19]. Throughout this time however, 'The Zodiac' would continue to write letters and postcards to a number of newspapers across California claiming a body count of up to 37 murders by the end of correspondence in January of 1974.

Included in a number of these letters were ciphers, messages encoded by 'The Zodiac' which may contain clues as to other victims, or even the identity of 'The Zodiac' themselves. There were a total of 4 ciphers included, generally known as the Z408, the Z340, the Z13 and the Z32, referring to the number of characters included in each. The first cipher, the Z408, has been solved, however the further 3 have yet to be solved, further investigation into the ciphers is shown in section 2.1.

## 1.2. The Problem

The problem encountered is that where the Z408 was solved at the time of the original investigation, the Z340 cipher has yet to be solved despite a large amount of study of the cryptogram, by both academics and amateurs. During the last 40 years, the attempts to solve the Z340 have assumed that the cipher is a 'homophonic' cipher, as the Z408, a 'polyalphabetic' cipher, a 'double transposition columnar' cipher and a 'one time pad'. However none have been able to provide a suitable solution. Other attempts have attempted to find solutions through use of anagrams and arbitrary expanding letters into full words. A number have even attempted to simply to fit their solutions to predetermined messages[17].

More Recently a number of ideas have surfaced which could create a solution to the problem. These include the ideas that the cipher was actually written backwards, or at a rotation of 90°, 180° or 270[20]. The idea has even been posed that the cipher is actually meaningless, in an attempt to confuse. However, these ideas have also yet to be validated.

## 1.3. Aims

The main aim of my research and software development is to attempt to solve the Z340 cipher through use of a genetic algorithm. By creating a framework to create a genetic algorithm which can attempt to solve substitution ciphers, I hope to then use the frame work to attempt to solve the Z340 cipher. Extrapolating that the Z340 cipher is also a homophonic substitution cipher, like the Z408 cipher, I can attempt to create a solution for both ciphers, as well as a simple substitution cipher. This means I can refine the method I wish to use to attempt the Z340 cipher, by first working on the substitution cipher, and then the Z408 cipher, both of which I already have the solution to.

# 2 Background

## 2.1. Substitution Ciphers

### 2.1.1. Overview

Due to the assumptions made about the ciphers to make this a feasible investigation into the Zodiac case with genetic algorithms, there are two types of cipher which must be looked into. The simple substitution cipher, as a benchmark for investigation with the Z408 cipher, assumed to be a homophonic substitution cipher. The assumption is made that the Z340 cipher is also a homophonic cipher. It seems unlikely that the writer of the ciphers would be able to implement more than one complicated type of cipher, especially considering the mistakes he made in implementation.

The definition of a substitution cipher in cryptography is one where the letters of the alphabet are replaced by some standard method with another 'alphabet' converting a 'plain text' into a 'cipher text'. This new alphabet can be any collection of letters, numbers, lines and dots or any other written symbol. Letters can be replaced individually or in groups of any number of letters. The standard process by which letters are replaced is what decides what permutation of substitution cipher it is. A cipher which replaces one letter with another symbol is a 'simple substitution' cipher, one that works on groups of letters is a 'polygraphic' cipher. One which has a fixed alphabet size the same as the plain text alphabet is known as a 'monoalphabetic' cipher, and one which uses multiple cipher text alphabets is known as a polyalphabetic cipher[22]. A Homophonic cipher is one where single plain text characters map to more than one cipher text character in one alphabet.

What follows is more information on the two types of substitution cipher used in this investigation, the simple substitution cipher and the homophonic substitution cipher.

### 2.1.2. Simple Substitution Cipher

As previously mentioned, the simple substitution cipher is one where each plain text letter is replaced by another symbol. Each letter of the alphabet is associated with the letter of a second alphabet[21]:

| | |
|---|---|
| Plain text letters: | abcdefghijklmnopqrstuvwxyz |
| Cipher text letters: | ZYXWVUTSRQPONMLKJIHGFEDCBA |

Convention says that the plain text should be written in lower case, and the cipher text in capitals, or in this case small capitals.

The cipher alphabet is usually rotated, reversed or scrambled in a more complicated way. In the case above, the alphabet has been reversed, making it an 'Atbash' cipher. If the alphabet is rotated it is referred to as a 'Caesar' cipher. Something like this:

| | |
|---|---|
| Plain text letters: | abcdefghijklmnopqrstuvwxyz |
| Cipher text letters: | LMNOPQRSTUVWXYZABCDEFGHIJK |

The more complicated scrambling usually involves the use of a key word, which has repeated letters removed, and then has the remaining alphabet letters added in the usual fashion. But the alphabet can be scrambled in any number of more complicated ways, for which only the sender, and the intended recipient would have the key.

If we take the second example of a cipher alphabet and encrypt a set of words, say 'Simple substitution cipher', this will give us the cipher text:

simple substitution cipher.
DTXAWP DFMDETEFETZY NTASPC[18].

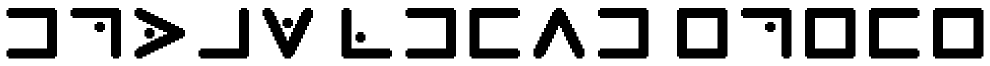
This provides us with an encrypted message, however it is obvious where words end, and punctuation gives away where the sentence ends, therefore punctuation would be removed, and the cipher would be split into uniform 5 letter groups, giving:

DTXAW PDFMD ETEFE TZYNT ASPC

All of this however does not create a particularly secure cipher. Though brute force is still not a good option, due to the fact that with a 25 letter alphabet, that creates a key space of 403,291,461,126,605,635,584,000,000[21] which is far too large to check every permutation in real time. However, simple substitution ciphers can be solved easily with the method of frequency analysis[21]. This is the process of finding patterns in the frequency of characters in a cipher, for example, the letter 'e' is the most common letter in the English language, appearing 11.1607%[22] of the time. From this we can expect that in any cipher, of long enough length to be statistically significant, the letter which appears approximately 12.228% of the time is the letter 'e'[30].

As well as simply looking for single letters, the English language has many sets of letters which often appear together, for example the collection of letters 'th' appear very often. The frequency of these groups of letters are described using n-grams. An n-gram is a way of storing an object as a vector[24], mostly used for analysing language, they can be used for words, syllables or letters. Analysis of the frequency of these n-grams can also be useful in the cryptanalysis of substitution ciphers.

Though the examples here have been shown using the Latin alphabet, the cipher alphabet can be any set of symbols, for example the pigpen alphabet, which was devised using lines and dots on a grid which looks something like this[18]:

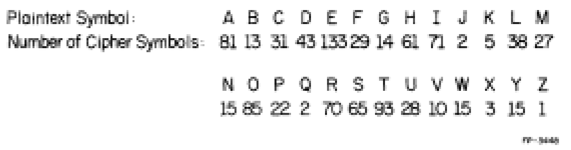


### 2.1.3. Homophonic Substitution Cipher

A homophonic substitution cipher is a substitution cipher in which a given character may have any number of different representations[25]. This allows the encoding to disguise the structural properties of the plain text and reduce the effectiveness of frequency analysis in the cryptanalysis of this type of cipher by making the single letter frequency of the cipher text constant. The elimination of frequency analysis as an efficient method of solving the cipher fixes one of the major problems with the simple substitution cipher. This use of multiple cipher text characters creates a key which looks something like this[25]:

| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C | R | Y | P | T | O | G | R | A | M | 5 | 6 | 7 | 8 | 9 | B | D | E | F | H | I | J | K | L | N | Q |
| 1 |  |  |  | 2 |  |  |  | 3 |  |  |  |  |  | 4 |  |  |  |  |  | S |  |  |  |  |  |
| U |  |  |  | V |  |  |  | W |  |  |  |  |  | X |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | Z |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

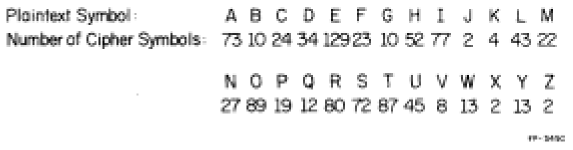
This example shown previously is not an effective homophonic cipher however, according to Fred Stahl in his paper "A homophonic cipher for computational cryptography"[25] to completely level the frequency range between letters you would require a homophonic cipher key described below.



| Plaintext Symbol: | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Number of Cipher Symbols: | 81 | 13 | 31 | 43 | 133 | 29 | 14 | 61 | 71 | 2 | 5 | 38 | 27 |
| | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| | 15 | 85 | 22 | 2 | 70 | 65 | 93 | 28 | 10 | 15 | 3 | 15 | 1 |

Figure 1 - Generalised homophonic substitution cipher to directly counter plain text letter frequency[25]

However, even with the use of multiple cipher text characters for each plain text letter, there are still problems which need to be taken into account. Bi-grams and tri-grams, two and three letter n-grams respectively, will still have frequencies which will be recognisable by a cryptanalyst. To counter this, the number of cipher text alphabet must again be altered to take into account these frequencies. By reducing the restriction on keeping the letter frequency curve flat, these other abnormal frequencies can be taken into account, resulting in a cipher like that shown below, also from Fred Stahl's paper.



| Plaintext Symbol: | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Number of Cipher Symbols: | 73 | 10 | 24 | 34 | 129 | 23 | 10 | 52 | 77 | 2 | 4 | 43 | 22 |
| | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| | 27 | 89 | 19 | 12 | 80 | 72 | 87 | 45 | 8 | 13 | 2 | 13 | 2 |

Figure 2 - Generalised homophonic substitution cipher[25]

Due to the large variation in cipher text letters, the length of the cipher now comes into play. The longer the cipher, the more of these variations can be used, therefore the more secure the cipher becomes.

## 2.2. Zodiac Ciphers

### 2.2.1. Overview

Between 1967 and 1974, the Zodiac killer provided more than 20 written communications to police officials, usually in the form of letters and postcards written to newspapers. However this also included letters to a prominent lawyer at the time, Melvin Belli, and to an independent TV station, KCAL-TV, channel 9. During the time the Zodiac was corresponding with the authorities, 4 ciphers were sent to the press, always contained within a larger covering letter, these are shown in Appendix A. The covering letters usually contained claims about his kills, threats about further attacks and demands on the public. The first cipher, the Z408, was sent as 3 different parts to the Vallejo Times-Herald, the San Francisco Chronicle, and the San Francisco Examiner. The further three were all sent to the San Francisco Chronicle on the being the Z408, the Z13 and the Z32.

Of the three, the only one to be solved was the Z408, it was solved by Donald and Bettye Harden less than a week after it was received by the newspaper. Other than this however only the Z340 has any possibility of ever being solved. Both the Z13 and the Z32, containing 13 and 32 characters respectively, have too few characters for a statistically significant solution to be found, there are too many possible combinations of characters. The fact that the smaller ciphers are unlikely to be solved has not however stopped people attempting to propose keys for the ciphers, however none have ever been verified.

### 2.2.2. Z408 Cipher

Received on the 31st July, 1969, the Z408 cipher is the longest and the only solved cipher sent by the Zodiac, while he was active. It was sent as three separate sections to three different newspapers:

1. Figure 3 - Part 1 received by the Vallejo Times-Herald
2. Figure 4 - Part 2 received by the San Francisco Chronicle
3. Figure 5 - Part 3 received by the San Francisco Examiner

The cipher was solved by a married couple of teachers, Donald and Bettye Harden[17], less than a week after the ciphers were originally published, and the solution published in the San Francisco Chronicle on 9th August, 1969. However it recently came to light that the authorities also received a solution to the cipher from a person claiming to only be a 'concerned citizen' on 10th August, 1969. Described as "substantially accurate"[26] by the FBI, it is still unknown who sent this second solution. All three parts of the cipher are shown below.

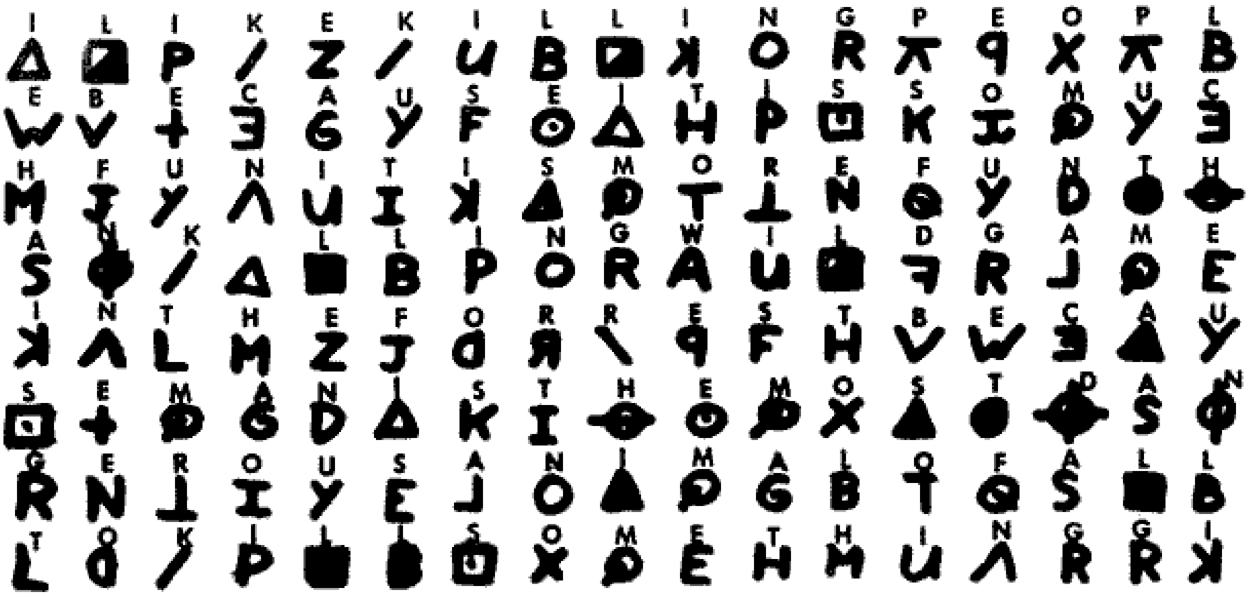

I L I K E K I L L I N G P E O P L
E B E C A U S E I T I S S O M U C
H F U N I T I S M O R E F U N T H
A N K I L L I N G W I L D G A M E
I N T H E F O R R E S T B E C A U
S E M A N I S T H E M O S T D A N
G E R O U S A N I M A L O F A L L
T O K S O M E T H I N G G I

Figure 3 - Z408 Cipher Part 1[19]

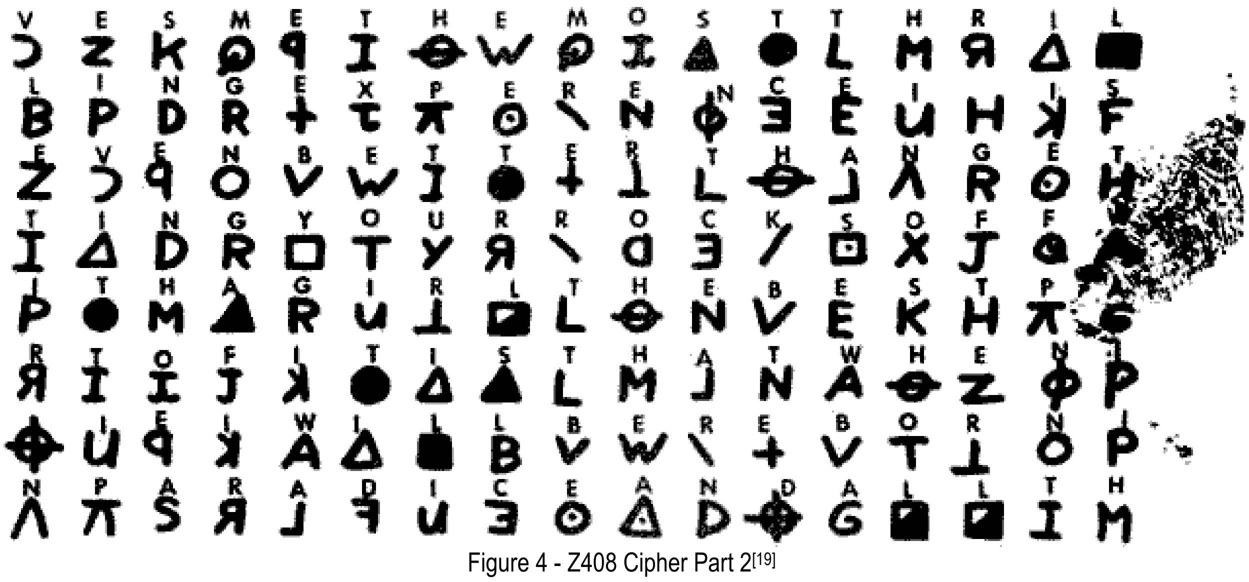

V E S M E T H E M O S T T H R I L
L I N G E X P E R E N C E I T I S
E V E N B E T T E R T H A N G E T
T I N G Y O U R R O C K S O F F
T H A G I R L T H E B E S T P
R T O F I T I S T H A T W H E N I
E I L W I L L B E R E B O R N
N P A R A D I C E A N D A L L T H

Figure 4 - Z408 Cipher Part 2[19]

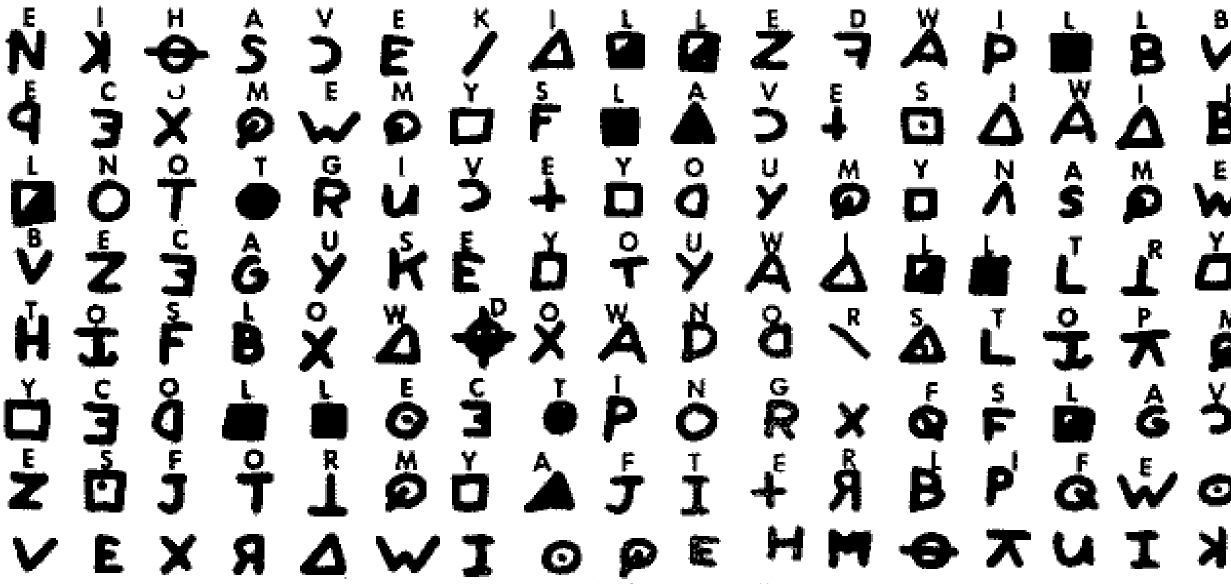


Figure 5 - Z408 Cipher Part 3[19]

The solution proposed by the Hardens is shown below, with the original spelling mistakes left in. A number of reasons for these errors have been proposed, ranging from the Zodiac losing his place in the enciphering process, using the wrong symbols intentionally to confuse, or simply the fact that the Zodiac was not a particularly good at spelling. Any one of these explanations could be true, or any combination thereof.

"I LIKE KILLING PEOPLE BECAUSE IT IS SO MUCH FUN IT IS MORE FUN THAN KILLING WILD GAME IN THE FORREST BECAUSE MAN IS THE MOST DANGEROUE ANAMAL OF ALL TO KILL SOMETHING GIVES ME THE MOST THRILLING EXPERENCE IT IS EVEN BETTER THAN GETTING YOUR ROCKS OFF WITH A GIRL THE BEST PART OF IT IS THAE WHEN I DIE I WILL BE REBORN IN PARADICE AND ALL THEI HAVE KILLED WILL BECOME MY SLAVES I WILL NOT GIVE YOU MY NAME BECAUSE YOU WILL TRY TO SLOI DOWN OR ATOP MY COLLECTIOG OF SLAVES FOR MY AFTERLIFE. EBEORIETEMETHHPITI"[17]

### 2.2.3. Z340

Following the Z408, the Z340 cipher was sent to the San Francisco Chronicle and published on 8th November, 1969. Unlike the Z408 cipher, this was sent in its entirety to a single newspaper. Though there have been many attempts  to solve the cipher, none of the proposed solutions have ever been verified, even up to the present day. Attempts have been made based on existing accepted cryptology documentation, but others have attempted to solve the cipher in unconventional methods, some even think the cipher was never meant to be solved and is merely meaningless. The entire Z340 cipher is shown over the page.

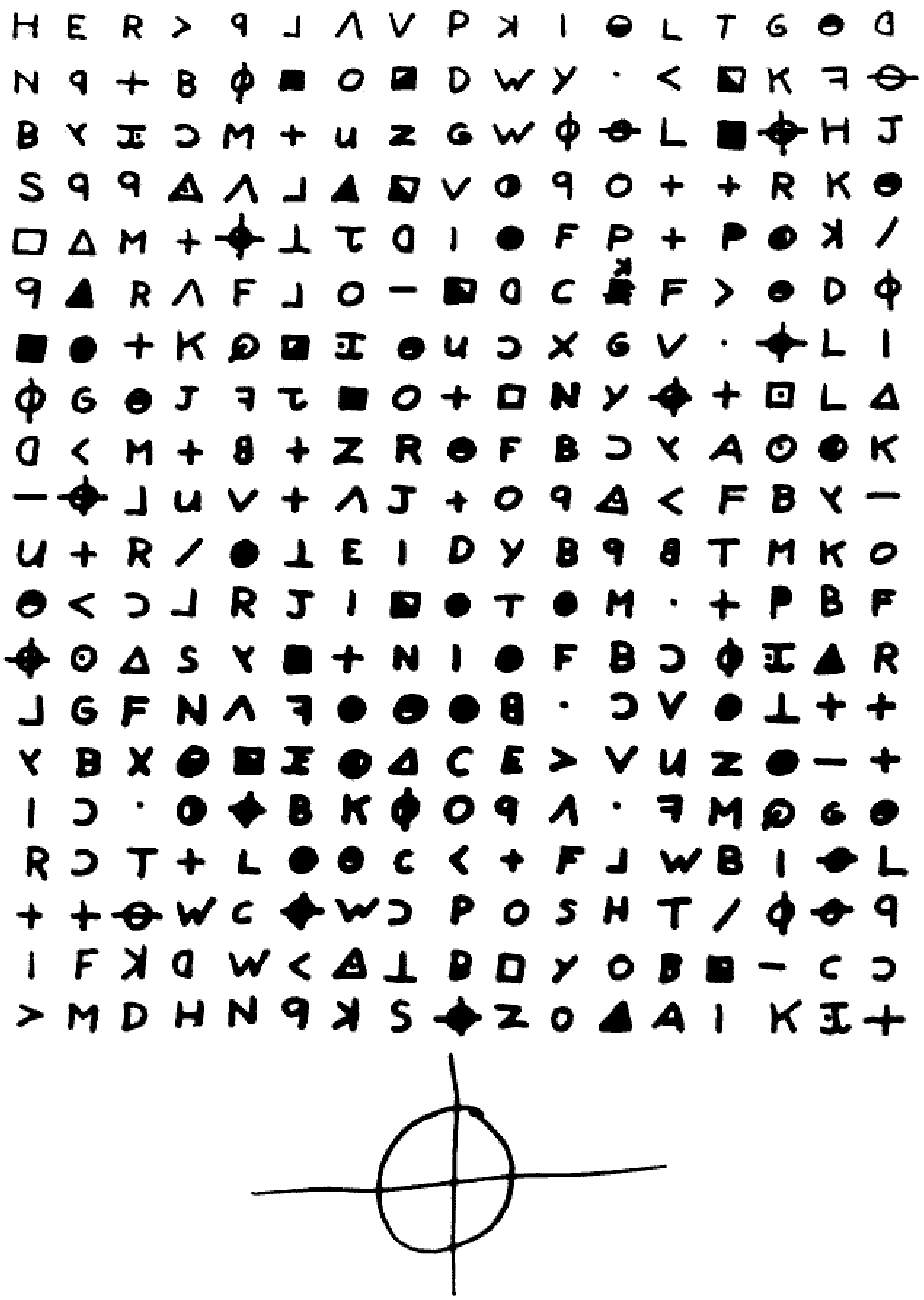


Figure 6 - Z340 Cipher[19]

#### 2.2.4. Z13 Cipher

Received and again published in the San Francisco Chronicle on the 20th April 1970, the Z13 is the shortest of the 4 ciphers, and was included in a letter which hinted at this cipher containing the Zodiac's real identity. The cover letter contained the line "My name is"[19] above the cipher itself, and with the length of the cipher this has led to many suggestions that the cipher is actually made up of the sentence "My name is" followed by the identity of the killer. Though this is one interpretation of the cipher there have been a number which have taken different ideas in order to find a solution.

However, due to the length of the cipher, it is unlikely any solution will be verified.

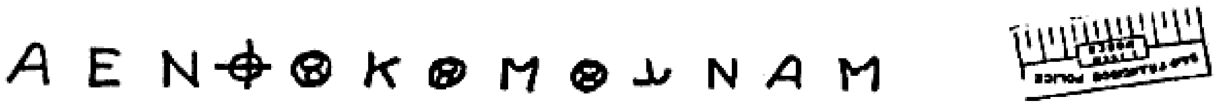


Figure 7 - Z13 Cipher[19]

#### 2.2.5. Z32 Cipher

The final cipher, received and published in the San Francisco Chronicle on 26 June 1970 was the Z32 cipher. This cipher could be something to do with the location of a bomb the Zodiac threatened was set, and would go off the following Autumn, or could be something to do with his annoyance with people who were not complying to his previous demands. But again, due to the length of the cipher, it is unlikely that any solution will be verified.

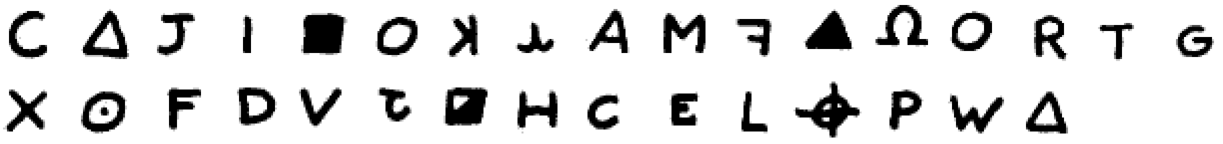


Figure 8 - Z32 Cipher[19]

### 2.3. Genetic Algorithms

#### 2.3.1. Overview

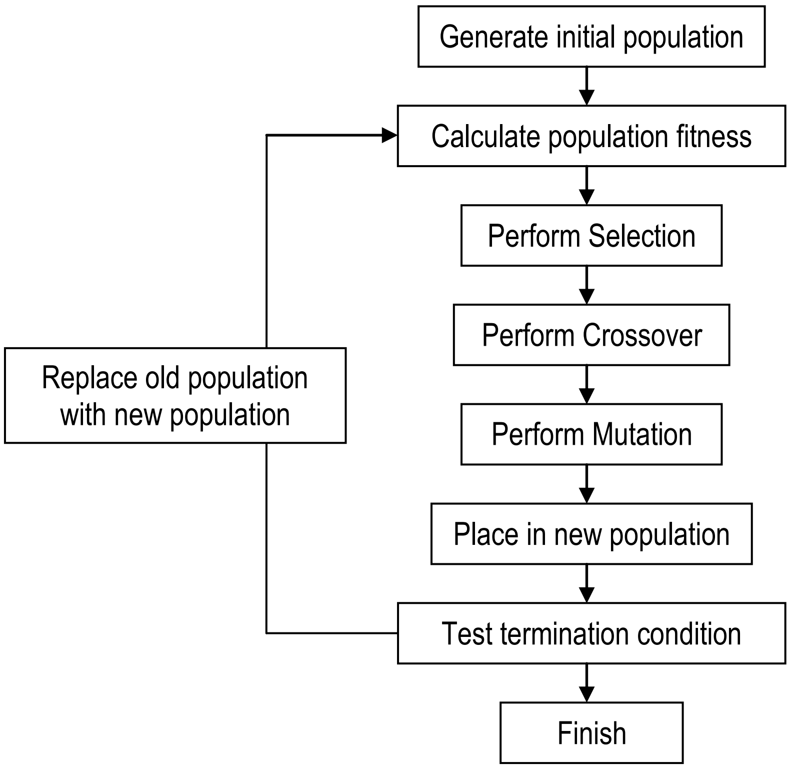
The genetic algorithm, is a subsection of an area of artificial intelligence called evolutionary computing. This area of artificial intelligence is interested in taking inspiration from the natural world, genetics and the ideas of natural selection and was first introduced by I. Richenberg in his 1960 paper "Evolutionary strategies"[28]. The definition of natural selection is the process whereby organisms in the natural world are more likely to survive and breed if they are better adapted to their surroundings, meaning the genes for the better adaptations are passed on to the next generation. Genetic Algorithms attempt to take these ideas and use them in the area of computer science. The terminology revolving around the ideas have been taken directly from the biological science, therefore there are a few terms which require definitions.

- Population - A group of candidate solutions, each individual is represented by a chromosome.
- Chromosome - A member of a population, an encoding of the solution made up of a series of alleles. Often represented by a binary string.

- Allele - A single piece of information about a solution. In the example of a binary string, each binary digit would be an allele.
- Fitness - The measure of how close a solution is to being correct. In biological terms, how adapted an individual is to its surroundings.
- Selection - The process of choosing those solutions in a population which will be allowed to reproduce to create the next generation.
- Reproduction - The process by which members of the current population interact and are altered in order to create the new generation population. Made up of a number of genetic operators, usually crossover and mutation.
- Crossover - The process of creating 2 offspring by combining the alleles of the current population.
- Mutation - The process of randomly altering a single chromosome in order to help the algorithm not to stall in at a local high fitness, rather than the global high fitness.

The overview of the genetic algorithm is really quite simple, consisting of a generation of a randomised population at the commencement of the algorithm then simply looping through the process of selecting a new population, performing reproduction and then calculating the fitness of the new populations. This looping process will be repeated until a certain termination condition is completed, which could simply be a time limit or a certain number of populations, or it could be something more complicated like calculating whether the fitness of the population is still improving after each generation.

This outline of the process looks something like this:


Generate initial population
Calculate population fitness
Perform Selection
Perform Crossover
Perform Mutation
Place in new population
Test termination condition
Finish
Replace old population with new population

Each of these sections is covered in slightly more detail in the following sections, population generation in 2.3.2, fitness in 2.3.3, selection in 2.3.4, Crossover and Mutation in 2.3.5.

The most common use of a genetic algorithm is for an optimisation problem, though the range of what is being optimised can be very large. For example genetic algorithms have been used to optimise a paper production process, or to optimise the schedule of production lines, and even optimise the design of engines for commercial passenger aircraft[9]. The use of genetic algorithms has become very widespread in a number of industries. In the case of this research it could be said that we are trying to optimise the key for a cipher, rather than having to attempt every possible permutation in a brute force attack.

The genetic algorithm can however also be used in conjunction with other areas of artificial intelligence. A specific example would be that a genetic algorithm can be used in both the construction and training of artificial neural networks[10].

### 2.3.2. Population Generation

Population generation and the creation of chromosomes, also referred to as problem encoding, is one of the 2 areas of a genetic algorithm which are problem specific[6]. The other being the fitness evaluation function.

The aim of encoding the problem in chromosomes is to contain some information about the solution it represents. The most used encoding of a chromosome is a binary string[28], which could look something like this:

| | |
|---|---|
| Chromosome 1 | 1101100100110110 |
| Chromosome 2 | 1101111000011110 |

Figure 9 - An example of chromosomes encoded as binary strings[27].

Each of these chromosomes could represent the solution in a number of ways, each bit of the string could represent some binary characteristic of the solution, or the whole string could represent a number. There are disadvantages to all actions however, and in this case for example, using a bit string to represent a decimal number, the numbers 15 and 16 are neighbours in integer space. However in the binary representations, 01111 and 10000, are not neighbours in the bit space. This could create problems in terms of attempting combination in crossover[10]. Problems arise in the case of a problem with a discrete number of solutions. If there are exactly 1200 solutions to a problem, this requires a bit string of at least 11 bits to cover. However this leaves a range of 2048 possible encoding, 848 bit patterns which are not used. This can be solved using a default worst possible encoding, or maybe encoding a more likely solution multiple times, to increase likelihood[6].

The binary string is not the only way to encode. Chromosomes could take almost any encoding, depending entirely on the problem which is being solved. This could include anything from encoding actual integer values, to encoding the problem in tree form[28].

The initial population generation is one of the more important features of any genetic algorithm, and there are a large number of factors which could be taken into account and could have an influence on the success of the genetic algorithm. Some of these factors could be the fitness function, the number of individuals, the problem difficulty, the diversity of the population and the search space[14]. Any one of these factors could heavily influence how the initial population is created, or they could simply be taken into account, for example if the population is created in a random, or pseudorandom fashion.

It is also shown that the creation of a good initial population has a positive effect on the likelihood that there will be a good solution found, as well as the reverse , that a bad initial population makes it difficult for a good solution to be produced by the genetic algorithm[28].

Though the effect of population size on the results of a genetic algorithm are very subjective, experimentation by M. Odetayo into differing population sizes[15] has established some results countering previous work in the area claiming the most efficient initial population size was 100. Depending on the requirement of the genetic algorithm, a population size of 100 on average sampled the least amount of points, had the shortest possible time and number of generations, this population size also had the highest averages in terms of time taken and generations. A population size of 300 however had the shortest average time taken and number of generations required[15]. Again, as this data was taken from one specific use of a genetic algorithm, this data could still be very subjective depending on data used and the problem encoding.

### 2.3.3. Fitness

The second area of a genetic algorithm which will depend on the problem being assessed, the fitness function, will probably be the most complicated part of the algorithm. The assessment of the population can be done in many different ways, it could be that the success is simulated in some way, or that the evaluation may be performance based, representing only an approximation of how fit a solution is[6].

Though the fitness function of a genetic algorithm is very subjective and will reflect the problem, one thing a that will be constant will be the need for speed in processing. Due to what are effectively large population sizes, for which the fitness function must be run for each member of the population, the evaluation must be completed quickly. For example for a population of 200 strings, if evaluation takes an hour, it takes over a year to complete 10000 evaluations. This would only be approximately 50 generations[6].

### 2.3.4. Selection

Selection is the process by which the next generation is chosen, those which will then be put through the process of crossover and mutation. Darwin's theory of natural selection says that the best should always be those that survive to create offspring[28], however there are a huge range of methods by which the next generation of chromosomes can be achieved. There are three well known functions which I will focus on, these are roulette wheel selection, stochastic universal sampling and tournament selection.

Roulette wheel selection, or fitness proportionate selection, is the process by which chromosomes are chosen depending on the fitness of the population. The idea of this type of selection is that a random value is selected between 0 and the sum of the fitness of the population. This random value will refer to the sum of the fitness value up to a certain chromosome in the population[28]. Which means that there is a higher chance of selecting chromosomes with a higher fitness value, as they take up more of the total fitness value. More higher fitness value chromosomes will be selected, however more identical chromosomes will be selected. This can mean that the variation in the population will reduce rapidly, which can lead to a local maximum value unless mutation is high enough to counter this reduction in genetic variation. A small population size, or one chromosome with a very high fitness can be a serious problem with this type of selection.

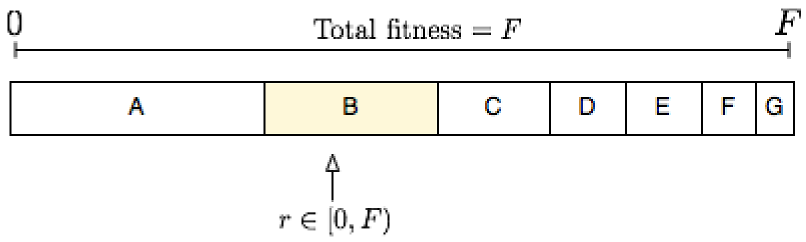


Figure 10 - A visualisation of Roulette Wheel Selection[32]

Stochastic universal sampling is a very similar selection process to roulette wheel selection. Again the process involves calculating the total fitness of the population. However with that value we now divide that total fitness by the amount of chromosome we want to keep, call this value 'n'. By taking a random value between 0 and 'n' we can then sample the population for the amount of chromosomes we need at equal intervals. This selection function gives more possibility of selecting the lower fitness chromosomes, which can in turn keep more genetic variation in the population. This can keep some good parts of a solution which happen to be contained within a less good chromosome in the population for crossover.

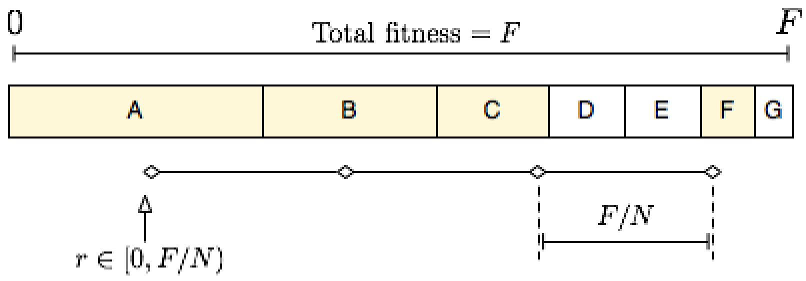


Figure 11 - A visualisation of Stochastic Universal Sampling[31]

The third process of selection, tournament selection is a different process of selection, which is extremely simple. The process involves taking a set number of chromosomes from the population, comparing them, and selecting the chromosome with the highest fitness. The process is repeated until enough chromosomes have been selected[29]. Tournaments of two chromosomes are those most used, however tournaments can be completed of any number of chromosomes. The larger the tournament size, the more often the quicker the average fitness of the population increases, but this also reduces the variation in the population more quickly[12].

Another idea which can be involved with the selection process is elitism. This is a process which can increase the success of a genetic algorithm by way of preventing the loss of the best chromosome[28]. Prior to the selection process the best, or a group of the best chromosomes, are added directly to the new population, then the selection process is completed in the normal fashion.

### 2.3.5. Genetic Operators

After the selection process there are two genetic operators which are completed, first crossover, followed by mutation. The process by which these are completed will change depending on how the problem is encoded, however the general ideas behind the processes will always be the same. Figure 12 shows how the population chosen by selection is then used to complete crossover to create the final population for the next generation.

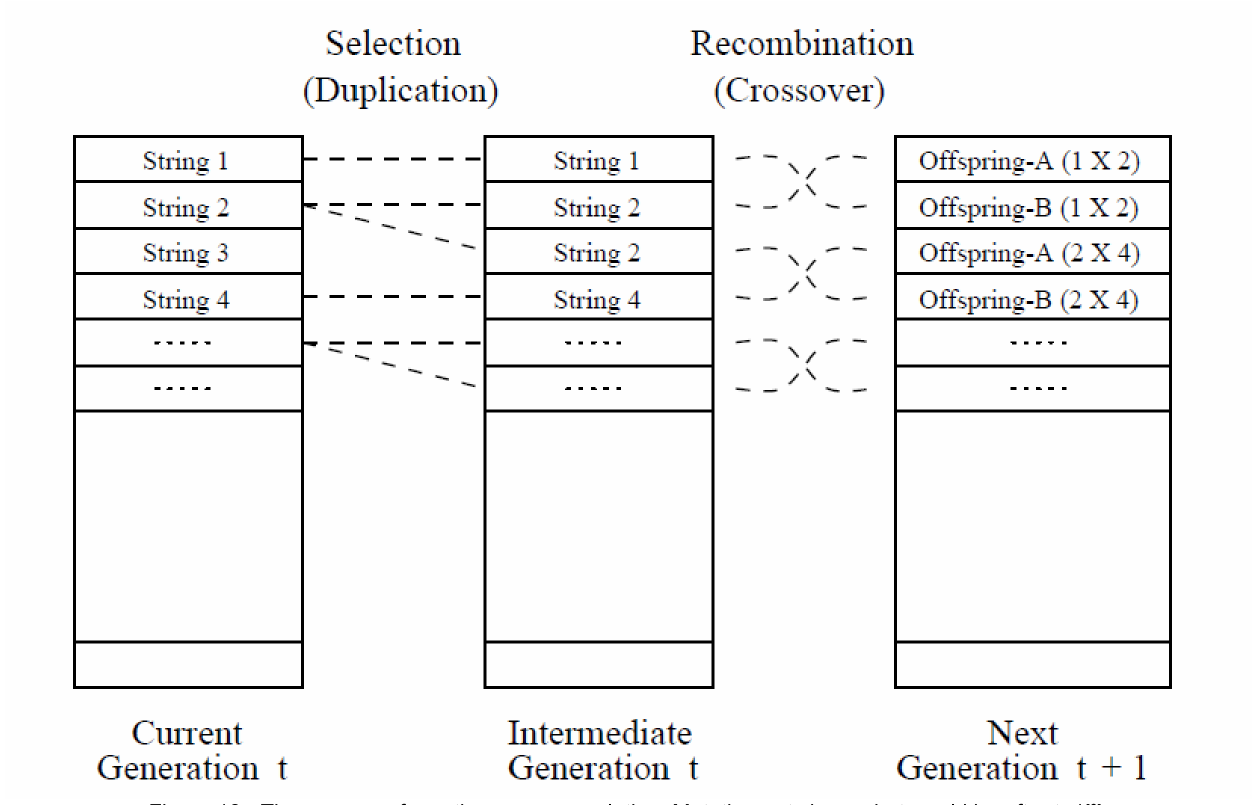


Figure 12 - The process of creating a new population. Mutation not shown, but would be after t+1[6]

Again using the example of a problem encoded using a bit string there are 4 general ways in which crossover can be completed single point crossover, two point crossover, uniform crossover and arithmetic crossover. Each of these methods creates a uniquely different way of varying the population of a genetic algorithm.

Single point crossover involves picking one specific point in the encoding of the chromosome. This could be a fixed point for every instance of crossover, or it could be a random point for every instance. The offspring is created by taking everything from before the crossover point from the first parent, and everything after the crossover point from the second parent[28].

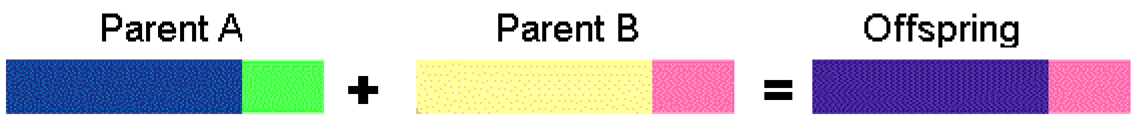


Figure 13 - A visualisation of single point crossover[27]

Dual point crossover is similar to single point crossover, however there are two crossover points selected. Everything from before the first crossover point is taken from the first parent, then everything from the first crossover point to the second crossover point is taken from the second parent. Finally everything from the second crossover point to the end of the chromosome is taken from the first parent again[28].

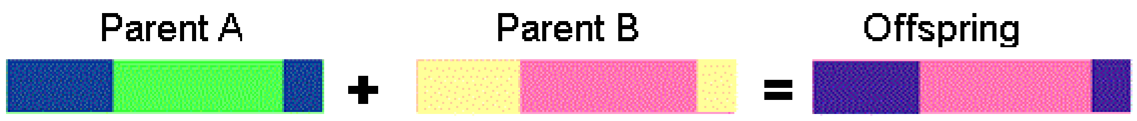


Figure 14 - A visualisation of dual point crossover[27]

For both single point crossover and dual point crossover it is possible to take 2 offspring from each crossover combination, simply by reversing the parents in the process. For example, in single point crossover one offspring has the first part of parent one, and a second offspring has the first part of parent two.

The third type of binary crossover, uniform, involves taking random bits from either chromosome and inserting them into the other, creating a randomised new chromosome[28].

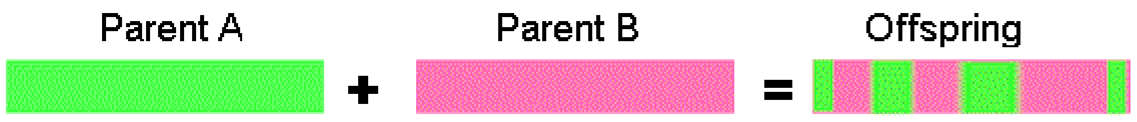


Figure 15 - A visualisation of uniform crossover[27]

The final type, arithmetic crossover, involves completing an arithmetic function to combine the two chromosomes[28]. In the case of a bit string this could be any sort of Boolean function, however any mathematical problem encoding could use a mathematical function, provided the output is always a valid encoding of the problem.

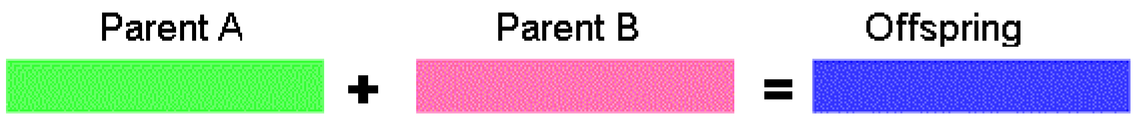


Figure 16 - A visualisation of arithmetic crossover using a Boolean function[27]

Mutation is the simplest function in the whole genetic algorithm, though it performs a very important task. In the example of a bit string, mutation can be seen in 2 ways. Firstly a random value can be assigned to the selected bit, which means that statistically the bit will only change 50% of the time[9]. The second method, visualised below, is that the selected bit is inverted, changing the bit every time[28].

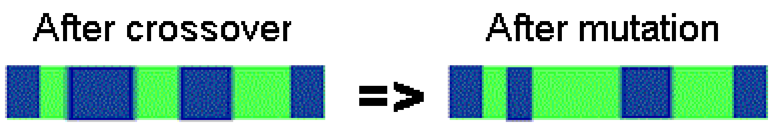


Figure 17 - A visualisation of bit inversion mutation[27]

The processes of crossover and mutation are governed by the crossover and mutation rates. These are the probabilities that crossover or mutation will occur for a specified chromosome. A higher crossover rate is usually applied, to ensures there is enough change between one generation and the next to make the algorithm worthwhile. The crossover rate is usually suggested at about 60%, however it can be raised to 80-95%[28]. Mutation rate however goes to the other extreme, the usual suggested mutation rate is between 0.5-1%[6][28]. The mutation rate however does need to be high enough to make sure that enough variation remains in the population to avoid local optimisation maximums.

Once the processes of fitness evaluation, selection, crossover and mutation are completed it is said that one generation of the algorithm has been executed.

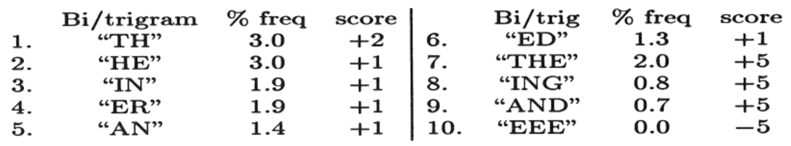
### 2.3.6. Genetic Algorithms for Cryptanalysis

There are a number of situations in which the genetic algorithm has been used in the area of cryptanalysis, showing that it is possible to attempt simple substitution ciphers[1][2][3], transposition ciphers[3][11], polyalphabetic ciphers[7][8], and even the Zodiac case itself[13]. Each of these investigations has taken different paths to attempt to solve substitution ciphers, but all but one is based around the archetypal genetic algorithm structure.

Though not all the attempts are successful, and there is a body of thought that for any but the simplest of implementations of cipher will be more quickly and easily solved by traditional cryptanalysis tools[3]. There have been some successes and useful information surrounding methods of creating genetic algorithms.

In the area of problem encoding, the general overview is that each chromosome will represent a possible key for the cipher. This ranges from a simple representation of a randomised alphabet[2], to representing the keyword of a Vigenere polyalphabetic cipher[8]. However the major consensus is that this is the most efficient way of encoding the solution.

In terms of attempts to create fitness functions for use in the genetic algorithm, there also seems to be consensus between the different investigations revolving around the use of statistics of the English language. This ranged from simple use of letter frequency comparison between the English language and the cipher text[2], to the more complicated use of the frequency of bi-grams, tri-grams and n-grams[7][8]. Even within the use of n-gram statistics there was variation in their use. The simplest approach was to take a small number of these n-gram examples and apply a arbitrary value to each one, only using the most frequent of n-grams, shown in Figure 18.



| | Bi/trigram | % freq | score | | Bi/trig | % freq | score |
|---|---|---|---|---|---|---|---|
| 1. | "TH" | 3.0 | +2 | 6. | "ED" | 1.3 | +1 |
| 2. | "HE" | 3.0 | +1 | 7. | "THE" | 2.0 | +5 |
| 3. | "IN" | 1.9 | +1 | 8. | "ING" | 0.8 | +5 |
| 4. | "ER" | 1.9 | +1 | 9. | "AND" | 0.7 | +5 |
| 5. | "AN" | 1.4 | +1 | 10. | "EEE" | 0.0 | −5 |

Figure 18 - N-gram scores for a fitness function[7]

Another example took a much more mathematical measure, shown in Figure 19, which is explained in the following fashion.
"Here, A denotes the language alphabet (i.e., for English, [A. . . Z], K and D denote known language statistics and decrypted message statistics, respectively, and the indices u, b and t denote the unigram, bigram and trigram statistics, respectively. The values of α, β and γ allow assigning of different weights to each of the three n-gram types"[8]

$$C_k = \alpha . \sum_{i \in A} \left| K^u_{(i)} - D^u_{(i)} \right| + \beta . \sum_{i,j \in A} \left| K^b_{(i,j)} - D^b_{(i,j)} \right| + \gamma . \sum_{i,j,k \in A} \left| K^t_{(i,j,k)} - D^t_{(i,j,k)} \right|$$

Figure 19 - Equation for calculating fitness based on n-gram statistics[8]

Selection, genetic operators and termination are generally dealt with in the traditional fashions, in all cases, except where the encoding of chromosomes is required to be a finite set of values, namely the Latin alphabet. This has required the slight alteration of crossover and mutation operators to ensure there are no repeated values and all chromosomes are valid sets of the alphabet[2][11]

## 3 Analysis

### 3.1. Z408

Due to the difficulty assessing or analysing either of the zodiac ciphers while still in the form written by the zodiac; and due to the large number of characters I had two choices as to the form of modelling I employed for the ciphers, I could either assign every different zodiac character a number, or I could assign each character an ASCII value[17]. I decided that the easiest way to convey the ciphers so that both humans and computers would be able to understand was to translate both ciphers into numeric form. This was an idea which has been employed by both prior works surrounding the Zodiac ciphers[4][13].

Table 1 shows the Z408 cipher transposed into numeric form, laid out in the same format as the cipher written by the Zodiac, 17 characters per line, over 25 lines. Work to show the development of this and the Z340 numeric ciphers is shown in Appendix B

Table 1 - Z408 cipher in numeric form

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 4 | 6 | 7 | 2 | 8 | 9 | 10 | 11 | 12 | 13 | 11 | 7 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 1 | 22 | 3 | 23 | 24 | 25 | 26 | 19 | 17 |
| 27 | 28 | 19 | 29 | 6 | 30 | 8 | 31 | 26 | 32 | 33 | 34 | 35 | 19 | 36 | 37 | 38 |
| 39 | 40 | 4 | 1 | 41 | 7 | 3 | 9 | 10 | 42 | 6 | 2 | 43 | 10 | 44 | 26 | 45 |
| 8 | 29 | 46 | 27 | 5 | 28 | 47 | 48 | 49 | 12 | 20 | 22 | 15 | 14 | 17 | 31 | 19 |
| 23 | 16 | 26 | 18 | 36 | 1 | 24 | 30 | 38 | 21 | 26 | 13 | 31 | 37 | 50 | 39 | 40 |
| 10 | 34 | 33 | 25 | 19 | 45 | 44 | 9 | 31 | 26 | 18 | 7 | 32 | 35 | 39 | 41 | 7 |
| 46 | 47 | 4 | 3 | 41 | 7 | 23 | 13 | 26 | 45 | 22 | 27 | 6 | 29 | 10 | 10 | 8 |
| 51 | 5 | 24 | 26 | 12 | 30 | 38 | 14 | 26 | 25 | 31 | 37 | 46 | 27 | 48 | 1 | 41 |
| 7 | 3 | 36 | 10 | 16 | 52 | 11 | 21 | 49 | 34 | 40 | 17 | 45 | 6 | 22 | 8 | 20 |
| 5 | 51 | 12 | 9 | 15 | 14 | 30 | 37 | 16 | 33 | 46 | 38 | 33 | 29 | 10 | 21 | 22 |
| 30 | 1 | 36 | 10 | 53 | 32 | 19 | 48 | 49 | 47 | 17 | 4 | 23 | 13 | 28 | 35 | 42 |
| 3 | 37 | 27 | 1 | 10 | 6 | 33 | 2 | 46 | 38 | 34 | 15 | 45 | 24 | 22 | 11 | 18 |
| 48 | 30 | 25 | 28 | 8 | 37 | 1 | 31 | 46 | 27 | 44 | 34 | 42 | 38 | 5 | 40 | 3 |
| 50 | 6 | 12 | 8 | 42 | 1 | 41 | 7 | 15 | 14 | 49 | 16 | 15 | 32 | 33 | 9 | 3 |
| 29 | 11 | 39 | 48 | 44 | 43 | 6 | 17 | 21 | 54 | 36 | 50 | 18 | 2 | 2 | 30 | 27 |
| 34 | 8 | 38 | 39 | 51 | 45 | 4 | 1 | 2 | 2 | 5 | 43 | 42 | 3 | 41 | 7 | 15 |
| 12 | 17 | 13 | 26 | 14 | 26 | 53 | 20 | 41 | 31 | 51 | 16 | 23 | 1 | 42 | 1 | 7 |
| 2 | 9 | 32 | 37 | 10 | 6 | 51 | 16 | 53 | 47 | 19 | 26 | 53 | 29 | 39 | 26 | 14 |
| 15 | 5 | 17 | 18 | 19 | 24 | 45 | 53 | 32 | 19 | 42 | 1 | 2 | 41 | 46 | 33 | 53 |
| 22 | 25 | 20 | 17 | 13 | 1 | 50 | 13 | 42 | 36 | 47 | 19 | 54 | 46 | 25 | 11 | 26 |
| 53 | 17 | 47 | 41 | 41 | 21 | 17 | 37 | 3 | 9 | 10 | 13 | 35 | 20 | 2 | 18 | 51 |
| 5 | 23 | 28 | 32 | 33 | 26 | 53 | 31 | 28 | 30 | 16 | 48 | 7 | 3 | 35 | 14 | 21 |
| 15 | 45 | 13 | 48 | 1 | 14 | 30 | 21 | 26 | 45 | 22 | 27 | 38 | 11 | 6 | 30 | 8 |

The  character frequency of the Z408 which is shown in Figure 20 demonstrates there is variance in the character frequency shown, however from prior research it is known that it is impossible to have a homophonic with a completely flat frequency curve. As well as this we know that this cipher is definitely a homophonic substitution cipher from the solution proposed by the Hardens.

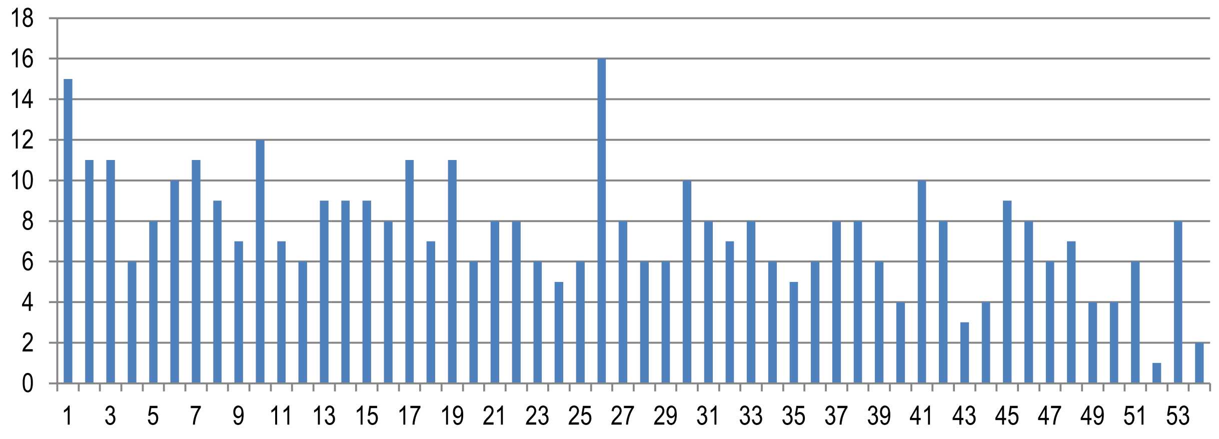
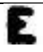
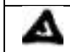
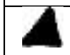
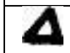
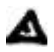


Figure 20 - Character frequency of the Z408 cipher

As well as the character frequency of the cipher, we can analyse the errors made by the Zodiac when writing the cipher in an attempt to find any advantage we can use. There are 6 cipher characters which have either been used in error, or have intentionally been used for multiple plain text letters. These are shown below in Table 2.

Table 2 - Z408 Character Errors[17]

| Zodiac Character | Latin Mapping | Appearances |
|---|---|---|
|  | E | 8 |
| | S | 1 |
| | T | 10 |
| | O | 1 |
| | E | 5 |
| | T | 1 |
|  | A | 1 |
| | S | 2 |
|  | A | 4 |
| | I | 1 |
| | S | 3 |
|  | I | 13 |
| | W | 1 |

From most of these errors I would conclude that the Zodiac has simply made mistakes, considering that there is usually only one erroneous value associated with the character. However, in the case of  and it may be that either the Zodiac was unsure of whether there was differing symbols, or may have simply mixed up what each symbol represented. As the most common characters represented by these two characters are 'a' and 's' it may be that one represents each of these characters but the Zodiac got confused between the two, and that the only serious error is that of using 'i' once, which would be in keeping with the errors which had been previously written into the cipher. Though this is an interesting observation of the cipher, it does not help with the problem of solving the cipher. If they are simply errors as I concluded it would be very difficult to take these into account in a genetic algorithm, and would simply have to be spotted in the cipher, if the genetic algorithm finds the correct key.

The total key space for the this cipher is $26^{54}$.

## 3.2. Z340

In Table 3, the Z340 cipher has been transposed into numeric form in the same way as the Z408 cipher.

Table 3 - Z340 cipher in numeric form

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 5 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 |
| 20 | 34 | 35 | 36 | 37 | 19 | 38 | 39 | 15 | 26 | 21 | 33 | 13 | 22 | 40 | 1 | 41 |
| 42 | 5 | 5 | 43 | 7 | 6 | 44 | 30 | 8 | 45 | 5 | 23 | 19 | 19 | 3 | 31 | 16 |
| 46 | 47 | 37 | 19 | 40 | 48 | 49 | 17 | 11 | 50 | 51 | 9 | 19 | 52 | 53 | 10 | 54 |
| 5 | 44 | 3 | 7 | 51 | 6 | 23 | 55 | 30 | 17 | 56 | 10 | 51 | 4 | 16 | 25 | 21 |
| 22 | 50 | 19 | 31 | 57 | 24 | 58 | 16 | 38 | 36 | 59 | 15 | 8 | 28 | 40 | 13 | 11 |
| 21 | 15 | 16 | 41 | 32 | 49 | 22 | 23 | 19 | 46 | 18 | 27 | 40 | 19 | 60 | 13 | 47 |
| 17 | 29 | 37 | 19 | 61 | 19 | 39 | 3 | 16 | 51 | 20 | 36 | 34 | 62 | 63 | 53 | 31 |
| 55 | 40 | 6 | 38 | 8 | 19 | 7 | 41 | 19 | 23 | 5 | 43 | 29 | 51 | 20 | 34 | 55 |
| 38 | 19 | 3 | 54 | 50 | 48 | 2 | 11 | 25 | 27 | 20 | 5 | 61 | 14 | 37 | 31 | 23 |
| 16 | 29 | 36 | 6 | 3 | 41 | 11 | 30 | 50 | 14 | 53 | 37 | 28 | 19 | 52 | 20 | 51 |
| 40 | 63 | 47 | 42 | 34 | 22 | 19 | 18 | 11 | 50 | 51 | 20 | 36 | 21 | 58 | 44 | 3 |
| 6 | 15 | 51 | 18 | 7 | 32 | 50 | 16 | 53 | 61 | 28 | 36 | 8 | 53 | 48 | 19 | 19 |
| 34 | 20 | 59 | 12 | 30 | 35 | 53 | 47 | 56 | 2 | 4 | 8 | 38 | 39 | 50 | 55 | 19 |
| 11 | 36 | 28 | 45 | 40 | 20 | 31 | 21 | 23 | 5 | 7 | 28 | 32 | 37 | 57 | 15 | 16 |
| 3 | 36 | 14 | 19 | 13 | 12 | 63 | 56 | 29 | 19 | 51 | 6 | 26 | 20 | 11 | 33 | 13 |
| 19 | 19 | 33 | 26 | 56 | 40 | 26 | 36 | 9 | 23 | 41 | 1 | 14 | 54 | 21 | 33 | 5 |
| 11 | 51 | 10 | 17 | 26 | 29 | 43 | 48 | 20 | 46 | 27 | 23 | 20 | 30 | 55 | 56 | 36 |
| 4 | 37 | 25 | 1 | 18 | 5 | 10 | 42 | 40 | 39 | 23 | 44 | 62 | 11 | 31 | 58 | 19 |

The character frequency of the Z340 has also been calculated in the same way, shown in Figure 21. From this we can see that the character frequency overall appears similar to that of a homophonic cipher, however there is one character which hugely outnumbers the other characters in terms of frequency. Character 19 ( ✚ in Zodiac alphabet) is very much an outlier in a type of cipher which is meant to reduce the cipher frequency curve. With a frequency of 24 it has double the frequency of the next most common character, which is character 20 ( B in Zodiac alphabet).

This oddity in the character frequency of the cipher pulls into question the validity of the assumption that the Z340 cipher is in keeping with the Z408 cipher, and is again a homophonic cipher. This makes character 19 one of the major problems with an attempt to solve the Z340, even if the assumption is still held that this is a homophonic cipher, the character needs to be taken into account. The idea that this character could represent a space between word[4] has previously been put forward. However I think this unlikely, due to the length of words this would create throughout the cipher. It is possible that the character could just represent some word spacing in the cipher, which would make the cipher easier to solve if removed. However I believe that if a genetic algorithm is able to find a key which is correct, or nearly correct, the fact that these letters are not correct should be obvious. The same applies to the idea that this character could be a meaningless character, placed in the cipher to confuse. In this case a visual inspection of a plain text created by an almost correct cipher key would be able to eliminate this character. With these assumptions, this character is left in the representation of the Z340 cipher for this investigation, and the Z340 cipher is still assumed to be a homophonic cipher.

The total key space for this cipher is $26^{63}$.

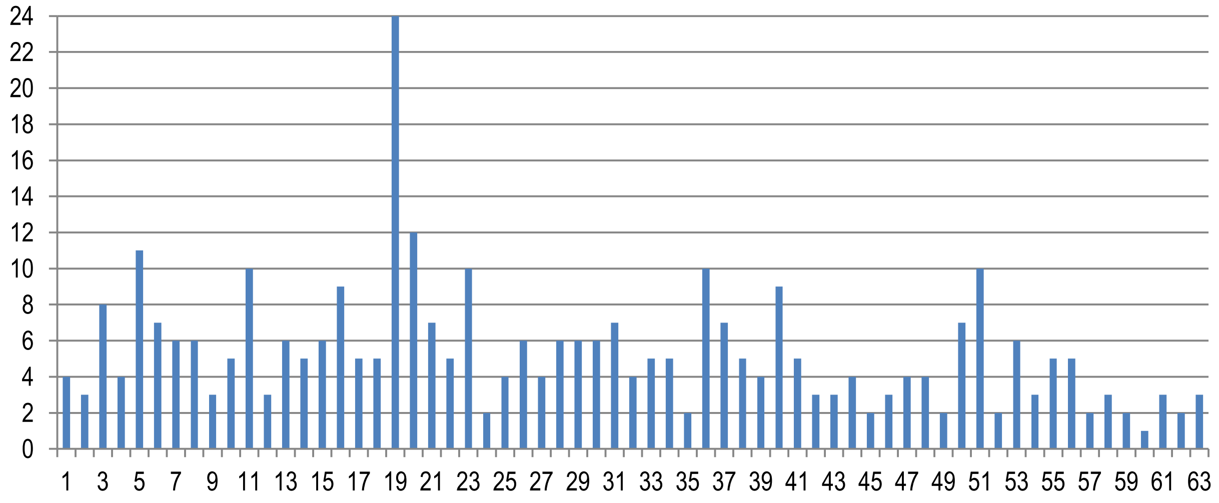


Figure 21 - Character frequency of the Z340 Cipher

### 3.3. Comparison of Z408, Z340 and Latin Alphabet

It is impossible to complete a direct comparison of the alphabets of the Z408 and the Z340. When the two alphabets are compared, there are 7 Zodiac alphabet characters used by the Zodiac in the Z408 cipher, which are then not found in the Z340 cipher, and there are 16 Zodiac alphabet characters in the Z340 cipher which are not found in the Z408 cipher. This means a direct comparison of characters is unlikely to result in the ability to find any data relevant to find the cipher key. The assumption is therefore made that the Z340 cipher key will have no association with the Z408 cipher key.

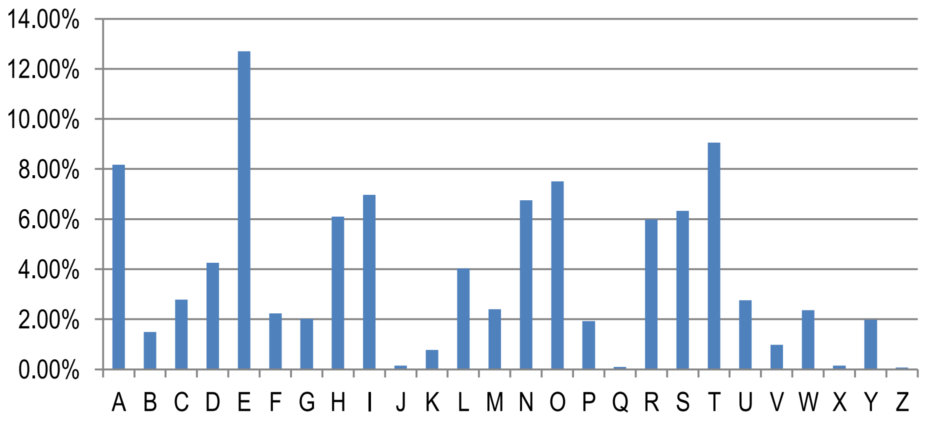


Figure 22 - English language letter frequencies[30]

The comparison of the Z340 cipher to the English language letter frequencies also confirms the assumption that the cipher is type of more complicated cipher. We can establish that with 63 Zodiac alphabet characters, there are more than 2 times the amount of characters than in the standard Latin alphabet. Looking at the frequencies of the English letters, the assumption can also be made that there has been an attempt to hide letter frequencies in the Z340 cipher, despite some oddities in the letter frequencies. This is in keeping with the assumption that the Z340 cipher is a homophonic substitution cipher.

One other way the letter frequencies of the ciphers, and the English language, can be compared is through the use of shannon entropy. The aim of shannon entropy is to quantify uncertainty, specifically, the average uncertainty in a random value. With the character frequencies shown previously the shannon entropy of English text, the Z408 cipher and the Z340 cipher can be calculated and are shown in Table 4[13].

Table 4 - Entropy comparison[13]

| | |
|---|---|
| English letters | 4.175 |
| Z340 cipher symbols | 5.72 |
| Z408 cipher symbols | 5.68 |

Both of the Z340 cipher and the Z408 cipher have entropies which are higher than that of the English language, which does not give much information. However the fact that the entropies of the Z340 cipher and Z408 cipher are so similar, only 0.04 apart, makes the fact that the ciphers are using similar encryption techniques much more likely, though it does not confirm it completely[13].

## 3.4. Genetic Algorithms

There were a number of options which could be implemented for the genetic algorithm, as previously noted. There are two options which are always going to be problem specific, these are the problem encoding and the calculation of fitness.

Firstly in problem encoding every one of the prior works has encoded the cipher key as their chromosome. Though this has changed for each implementation, and the chromosome will be specific to each cipher, this solidifies the most probable way to implement the problem encoding. The prior work also shows that chromosomes are usually encoded using bit strings. However most of the prior works are written using either prolog, or other specific artificial intelligence languages. The use of a high level object oriented language could offer some addition options on this front.

Secondly, in terms of fitness, most of the prior works use some sort of letter and n-gram frequency to calculate fitness values for their genetic algorithms. Due to the widespread use of these techniques in cryptanalysis tools which have been successful, it would be unwise to attempt to find a new way of assessing how good a plain text solution is. There have been both simple value based fitness assessments, and complicated equation based fitness assessments, both of which have been successful , and a method must be decided on which revolves around these ideas.

Thirdly both prior works[4][13] referring specifically to the Zodiac ciphers encode the original ciphers into a numeric form. As previously noted in section 3.1 and 3.2 this seems like a good way of analysing the ciphers. It also gives rise to the best way of modelling the ciphers for the genetic algorithm. The algorithm can be stored in some sort of matrix, which will allow the cipher text and the chromosome to be easily combined into a plain text, which can be used in calculating fitness.

For the remaining sections of the genetic algorithm, selection, genetic operators and termination, it is simply a case of deciding which of the functions from the wide number of existing functions will be used in this implementation. Those I have looked at in most detail in my background are the most likely candidates, as they give a range of functions with different benefits and drawbacks.

## 3.5. Development Tools

One major choice required at this point was the language used to implement this software. There are a number of languages specifically designed for artificial intelligence programming, for example Prolog, previously mentioned, and a number of other instances, specifically written for areas of artificial intelligence. However I had no prior knowledge of these language, and though a number of the prior works completed Prolog software, I thought that an object oriented language was more suitable for the modular design I wanted to develop. Considering this, I had experience with two languages which could have been used, Oracle Java, and Microsoft C# .NET. Considering that C# is an implementation of Java, there is not a huge amount of difference between the two languages, the only major one being the mobility of released code. C# is restricted to Microsoft OSs due to how the .NET framework is implemented. Due to the fact that I am more familiar with C# as a language and because I was not overly worried about the mobility of the software, plus I would be performing all research on a machine with access to a Microsoft Windows OS, I decided on a C# based implementation. Another point in this decision was that if I was ever actually required to move my implementation to a different OS, there is an open source development named Mono, which is developing a cross platform framework for C#[33], meaning I had a fall back point if required.

The use of C# meant that the official IDE Visual Studio was the obvious choice for development. This provided me with an easy to use environment which I was familiar with.

For a development process like this I wanted to be able to use a version control to keep track of development, as well as being able to control any problems I encountered. I had a choice of GitHub[34] or Apache Subversion[35] as version control software. I am more familiar with the functionality of Subversion, however without my own cloud based location to store this project, GitHub was the more useful choice. This allowed for cloud backup of my software, as well as the possibility of working from different locations.

# 4 Design

## 4.1. Process

Due to the idea of a modular application and the research focus on this project, I attempted to use a software development process which would provide structure while still providing enough flexibility for change while in the development process, in case of developments which could be included in the software for advantage. The process decided on was Feature Driven Development, mainly following "Feature Driven Development Processes", provided by nebulon.com, outlining the design and development process[5]. The separate development of features, and the process of designing high level, then focusing down on specifics, followed well how I wished to develop an overall model, followed by multiple modules.

## 4.2. Overview

Due to the format of a genetic algorithm lending itself to a modular implementation this made dividing the implementation easy. As previously mentioned, the genetic algorithm is made up of 7 major areas, a central management process, a method of generating a population, the chromosomes which make up the population, a method of calculating fitness, a method of selecting the next generation, a method of altering the population in a 'genetic' level and a function to calculate when to stop the algorithm. This presented a straight forward way of creating a high level design and framework which could then be used to generate multiple implementations of each module, allowing for investigation into differing methods of creating a solution, as well as allowing the application of the same modules to both simple substitution and homophonic substitution ciphers.

As I wished to create a modular object oriented implementation, I first designed a number of interfaces, which describes the simplest possible genetic algorithm functionality. This allows for modules to be added and used with existing code, as well as allowing generalised modules, such as selection and genetic operators , to function on any type of cipher based on the framework, thus reducing the amount of problem specific code required for the genetic algorithm to function properly. The framework is described in more detail in section 4.3. This framework of interfaces was then used to implement a number of each type of module, each of which is described in more detail in the following sections 4.4 to 4.11.

## 4.3. Features

The functionality of 6 features whose functionality must be implemented for this genetic algorithm to be successful. Some of these features could and will be implemented in a number of ways. Each feature will be explained here, and the variations of those which will be implemented in multiple ways will be presented. They are ordered with high-risk complicated features highest, preceded only by other features on which more complicated implementation will rely.

1. Population Generation - The implementation must have a method of generating a collection of individual cipher keys which could possibly be the solution to the problem. Included within this is the method by which the valid possible cipher keys are created. Chromosome structure and population generation are described further in 4.7 and 4.8 respectively.

2. Population Management - The implementation must have method of holding the previously generated and either performing actions on said population, or passing said

population to another feature to have actions performed on it. Population management is described further in section 4.6.

3. Fitness Calculation - The implementation must be able to perform a type of mathematical function on members of the population which will provide an arbitrary value which can be used to compare members of the population in terms of relative success decrypting the analysed cipher.
   - This will take the form of a normalised value created from a large selection of bi-gram and tri-gram values from a text file source. Different methods of fitness with be completed using differing combinations of source n-gram values. Methods of fitness are described further in section 4.9.

4. Genetic Operations - The implementation must be able to perform a set of crossover and mutation operations on the stored population as described previously in section 2.3.5.
   - The implementations of crossover will include both single point crossover and dual point crossover methods.
   - The implementations of mutation will include both an implementation assessing the probability of a single chromosome mutating, and assessing the probability of every allele in a chromosome mutating. Genetics are further discussed in section 4.10. Methods of mutation are further described in section 4.11 and crossover in section 4.12.

5. Selection - The implementation must be able to perform type of function which will pick a number of chromosomes from the current population and place them in another population.
   -The implementations of selection will include the processes of selection roulette wheel selection, stochastic universal sampling and tournament selection. Methods of selection are further described in section 4.13.

6. Termination - The implementation must be able to perform a type of calculation which will confirm when the algorithm is intended to complete.
   - The implemented method of termination will be based on time spent. Methods of termination are further described in section 4.14.

## 4.4. Overall Framework model

The feature driven development process calls first for an overall model of the implementation required. Interestingly this overall model very much resembles the plan for the interface model. It is similar to the high level design of what a genetic algorithm should look like, although it does not take into account any specific implementation. In this model a single interface, or a group of interfaces, implements part of the Features List, shown in the previous section.

1. Population generation is implemented by IPopulationGeneration and IChromosome. IChromomsome represents a simple data structure, all calculation and building of chromosomes is represented by IPopulationGeneration.
2. Population management is implemented by IPopulationManagement, implemented centrally, it associates with all other features.
3. Fitness calculation is implemented by IFitness and ICipher. ICipher acts as a data structure model of the actual cipher. All calculation is performed in IFitness.
4. Genetic operations are implemented by IGenetics, ICrossover and Imutation. IGenetics in overall control, calling ICrossover and IMutation to perform tasks.

5. Selection is implemented by ISelection.
6. Termination is implemented by ITermination.

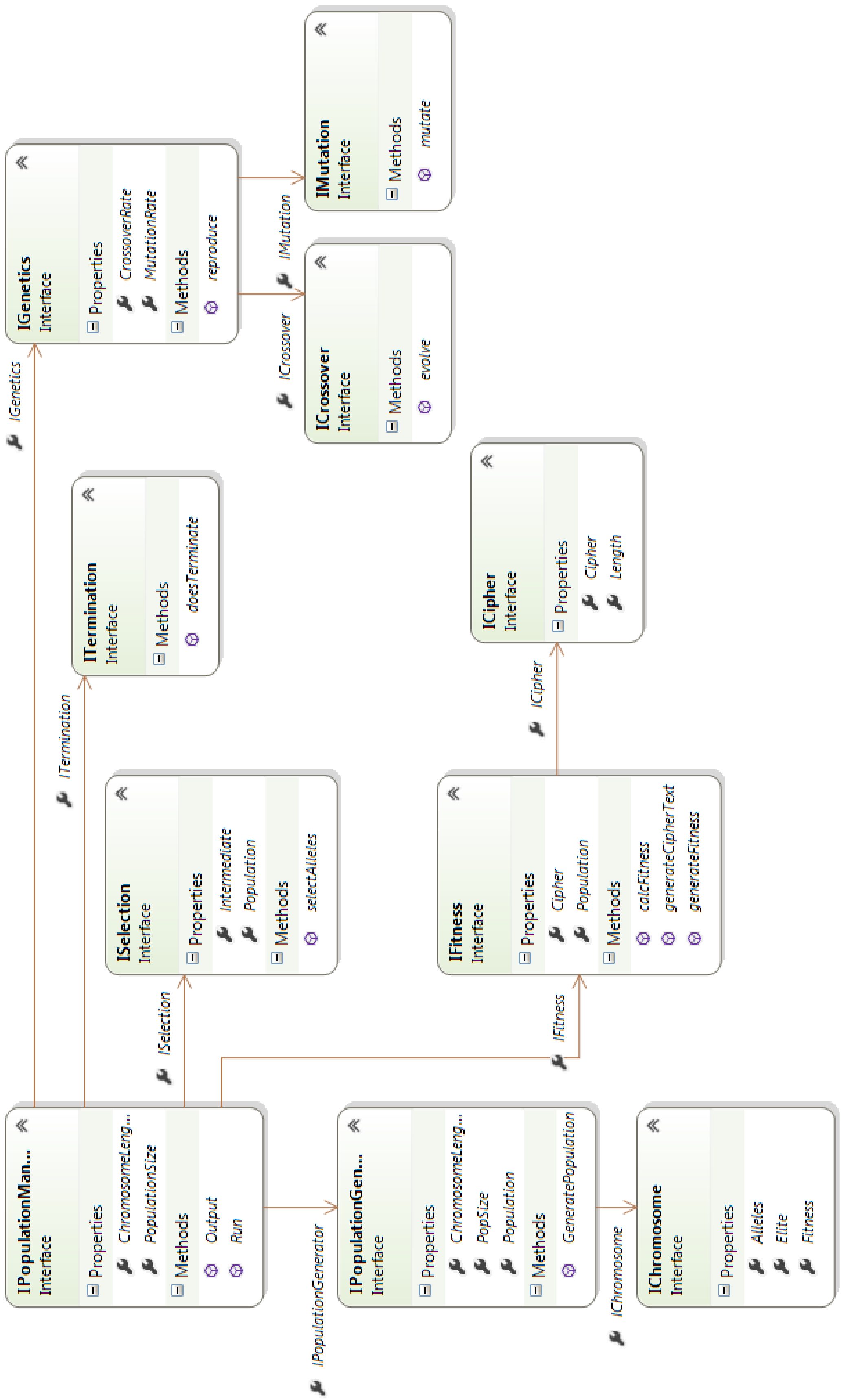
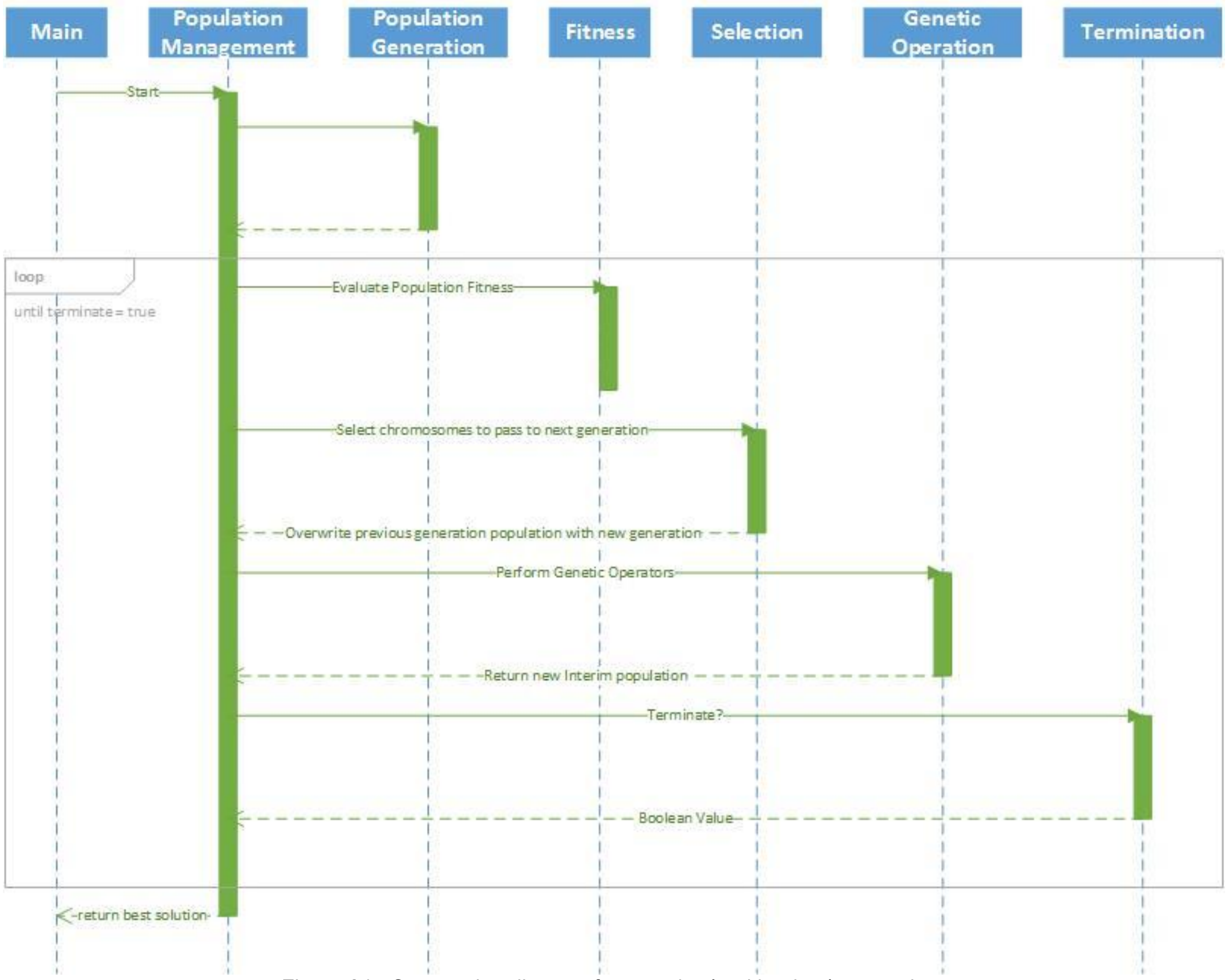


Figure 23 - Overall model with class associations and dependencies.

### 4.5. Sequencing

Based on the overview described, it is possible describe the algorithm in terms of these interfaces, where they will be called and how they will interact together. All of the features implementing part of the genetic algorithm will be self contained and will simply be provided with the current population, perform some sort of calculation or changes to the population and then pass the population back to the central population management. The process of passing the population, and completing the different operations is shown below in the sequence diagram of the algorithm which is to be implemented.



Main
Population Management
Population Generation
Fitness
Selection
Genetic Operation
Termination
Start
loop
until terminate = true
Evaluate Population Fitness
Select chromosomes to pass to next generation
Overwrite previous generation population with new generation
Perform Genetic Operators
Return new Interim population
Terminate?
Boolean Value
return best solution

Figure 24 - Sequencing diagram for genetic algorithm implementation

This implementation of the genetic algorithm allows for the maximum amount of variation in the implementation of each subsection of the overall process, while minimising the amount of dependency on the problem being assessed. For example as the structure of the chromosome is standardised, this minimises the necessity for problem specific code in the selection and genetics. Though depending on the data actually stored in the chromosome structure, there may need to be some variance.

To reiterate, this structure also gives the ability to swap out the methods of each of the operators. The application will be usable with the existing functionality written here, however with access to the source code it will be possible for anyone to add functionality to assess their own problem, or to assess an existing problem with a new implementation of a selection function or a genetic operator.

## 4.6. Management

The requirement of the management functionality has 3 main focuses. The first being taking the user input in terms of what operators the user wishes to use in the genetic algorithm, as well as which implemented cipher the genetic algorithm will investigate. Secondly it will act as the central control, managing what action is taken, in what order, and how long the algorithm should run for. Finally this part of the implementation will be required to create the output for the algorithm. To implement this part of the algorithm there will be a number of major design decisions which needed to be made.

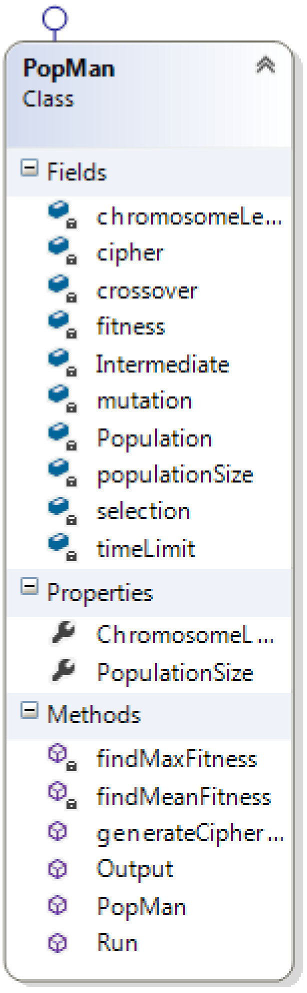


Figure 25 - Population management design

Firstly, the decision of whether to include a graphical user interface through which a user could make decisions about which parts of the algorithm to use. It would be possible to run this algorithm from the command line, however I considered that it would be clearer which options were possible through use of a GUI. Therefore this was the route I decided to take, planning a very simple GUI to complete all of the changeable options for the algorithm. The wireframe of the GUI is shown in Figure 26.

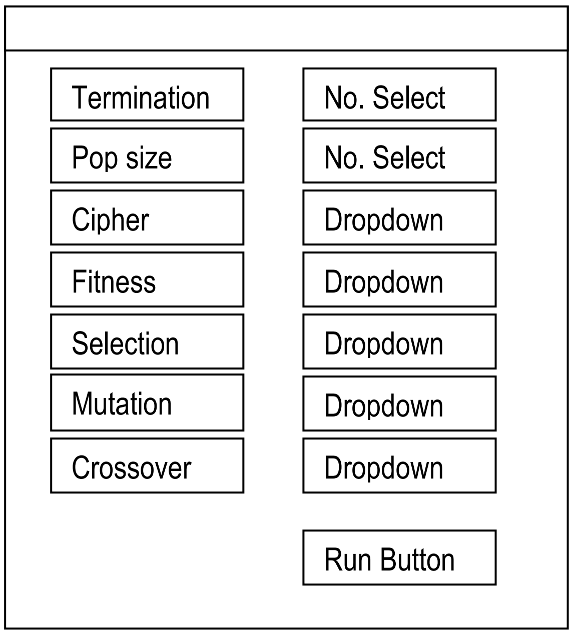


Figure 26 - GUI design wireframe

This GUI allows for the input in a simple, obvious method, and shows all of the possible selections for the genetic algorithm. The simplest possible way of transferring this data to the population management class would be the most effective. It may be possible to use reflection to add the algorithm options to the GUI, and to organise options within the population management. However the method settled on eventually was to use switch statements to complete this functionality. This would give the option of passing the necessary different options to each of the different classes, as well as being able to differentiate between substitution and zodiac specific functionality, while still showing the same options on the GUI for simplicity.

The second aim of the population management class is very simple. Using the termination class, a simple loop through the actions of calling different functionality until a flag is changed to show that the algorithm is stopped. The idea of this functionality is shown in the sequence diagram in Figure 24.

The third aim is to create the output of the algorithm. The simplest way of outputting the relevant data would be to output values to a text file. This is a straight forward function to complete and for the sake of analysing the results, it would be logical for each generation to output the best possible fitness in the generation, as well as the mean fitness of the whole population. Final output will be the cipher key associated with the maximum fitness, in case there is a decrease in fitness. This way it is still possible to see the best possible result found by the algorithm. For the final generation, I will also output the cipher, in full plain text, when decrypted by the best possible cipher key. This offers the chance to analyse the plain text visually, as it may be possible to improve the decryption of the key using the better pattern recognition of the human brain.

## 4.7. Chromosomes

The modular design of this genetic algorithm calls for the chromosome class to be considered as a simple data structure, simply containing information about a solution but not to perform any actual calculation, within the creation of a population or to perform any calculations upon the information stored within. This leads us to how the chromosome for a cipher key should be stored.

Considering that the information stored in the chromosome will be a key for a cipher, it can be assumed that the chromosome will require a list of representations of each of the cipher alphabet letters. For the Z408 cipher, this would be 52 items, for the Z340 cipher, 63. The easiest solution on this front would be to store a list of characters representing the interpretation of the cipher alphabet in the Latin alphabet. As it has been established that the ciphers will be stored in a numeric form, it will also be necessary to find a way of associating the cipher numeric representation with the character based cipher key. This problem is overcome due to the fact that it can be assumed that the index of the cipher key data structure indicated which cipher alphabet character is associated with the contained Latin alphabet character.

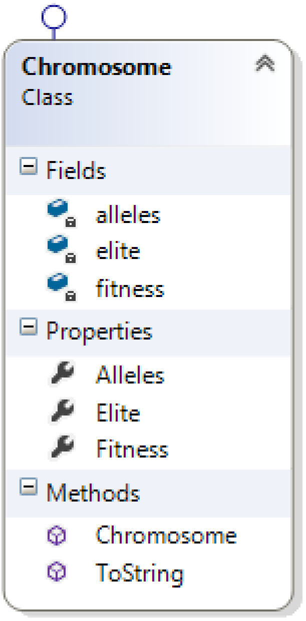


Figure 27 - Chromosome design

One minor side note is that if the same logic is used to represent the key for a substitution cipher, it will be necessary to store all 26 characters of the Latin alphabet. However the cipher may not contain every letter of the alphabet. This is a problem which will have to be solved when assessing the cipher in the fitness section of the genetic algorithm.

On top of the representation of the cipher key there are two other pieces of information which need to be stored within the chromosome. These are the fitness of the cipher key and, depending on the implementation of other parts of the genetic algorithm, the indication of whether a particular chromosome is considered elite.

The fitness of the chromosome could be anything from an integer to a decimal value, depending on how fitness is implemented. Therefore the value will have to be considered as some kind of floating point number, for which I settled on a double. From the design of the fitness functions I was implementing, there was no requirement for a data type with a larger range, precision.

The value of elite, will simply be a Boolean, and the storage of the value within the chromosome allows for the use of single or multiple elites within a population, depending on the needs of the algorithm implementation.

The UML description of the Chromosome data structure is shown in Figure 27, implementing the IChromosome interface. Due to the multiple pieces of information stored within the structure, it seemed logical to override the ToString method to contain all information pertaining to a chromosome.

## 4.8. Population Generation

### 4.8.1. Substitution Ciphers

Population generation is one of the parts of the genetic algorithm which is going to have to be problem specific. Due to the fact that I am going to attempt both the Zodiac ciphers and a substitution cipher, this means I will need two different methods of generating a population. Each would have to be able to create a population of a specified size with a randomized set of chromosomes for the specified problem.

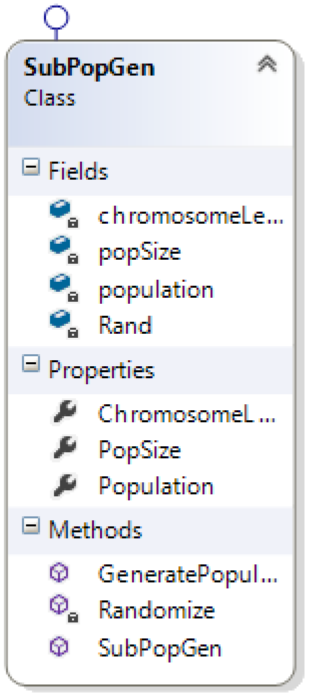


Figure 28 - Substitution cipher population generation design

The simpler of the two options is the substitution cipher key representation. In this case, the cipher key will always include 26 characters, and the set of characters will always be the same, the Latin alphabet. Even when a cipher is encoded with some other set of character for the cipher text, the key will always associate with the Latin alphabet.

To create multiple instances of valid cipher keys for the population there needs to be some way of shuffling an array containing the Latin alphabet characters, in some pseudorandom way, rather than attempting to create an array in a pseudorandom way. A simple method for completing this action is using a Fisher-Yates Shuffle[36]. This is an algorithm to randomly permute N elements by exchanging each element. This was useful in this case as it shuffled the set statically, without creating a new instance of the set. A sudo-code example of this algorithm is :

```
for i from 0 to n − 1 do
j ← random integer with 0 ≤ j ≤ i
if j ≠ i
a[i] ← a[j]
a[j] ← source[i]
```

This sudo random shuffle allows for the creation of any size population required by the user of the software. Figure 28 shows the UML description of how the substitution cipher population generation will be implemented. This design implements the IPopulationGeneration interface, but also uses added functionality, specific to this problem type.

### 4.8.2. Zodiac Alphabets

Despite the Z408 and the Z340 ciphers using differing alphabets, the same method of generating a population can be used for both due to the assumed similar enciphering methods. Assuming both are homophonic cipher, there is a necessity for repeated Latin alphabet characters within the cipher keys for each cipher. This means a sudo random method of assigning multiple instances of each Latin alphabet character to the cipher alphabets would be required. The method taken to implement this idea would be that letters can be assigned to the cipher alphabet in such a way that the frequency of letters follows the letter frequency of the English language. Hopefully even though the letter frequencies have been disguised within the cipher, it can be hoped that the letter frequencies are still generally followed within the text before it was enciphered.

Due to the complexity of the English language letter frequency values which were presented earlier in this document, it was important to simplify the values somewhat, to ensure that the implementation will create valid chromosomes, which are not too short or too long. With this in mind, a simplification of the English language letter frequencies to integers was devised. This gave a set of frequency values shown in table 5. Though this simplification hopefully improves the range of letters included in the cipher keys, it also removes the letters 'J', 'Q', 'X' and 'Z' from possibly appearing in the initial population. It is the hope that the infrequency of these letters in the language anyway will reduce any adverse effects this could have, as well as including these letters in mutation, allowing for the possibility they will appear in chromosomes while the algorithm is running.

Table 5 - Simplified letter frequencies

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| A | 8 | G | 2 | M | 2 | S | 6 | Y | 2 |
| B | 1 | H | 6 | N | 7 | T | 9 | Z | 0 |
| C | 3 | I | 7 | O | 8 | U | 3 | | |
| D | 4 | J | 0 | P | 2 | V | 1 | | |
| E | 13 | K | 1 | Q | 0 | W | 2 | | |
| F | 2 | L | 3 | R | 6 | X | 0 | | |

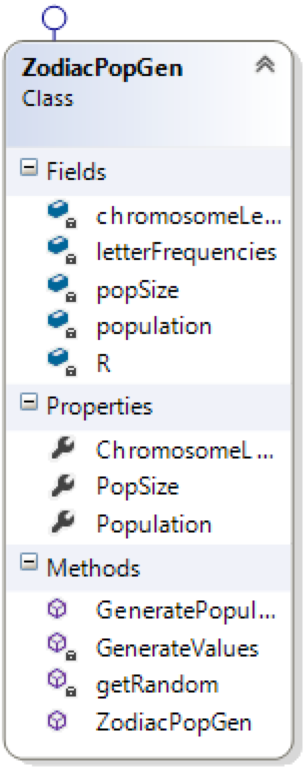

ZodiacPopGen
Class
Fields
chromosomeLe...
letterFrequencies
popSize
population
R
Properties
ChromosomeL ...
PopSize
Population
Methods
GeneratePopul...
GenerateValues
getRandom
ZodiacPopGen

Figure 29 - Zodiac cipher population generation design

Using these letter frequencies, randomised chromosomes are created by placing the correct number of each character in a random position throughout the chromosomes. This method of attempting to take into account letter frequencies will hopefully create better results than simply creating chromosomes with completely random characters.

This implementation of the Zodiac population generation, shown in UML form in Figure 29, implements the IPopulationGeneration interface as well. However it has different specific functionality to the substitution population generation, resulting in a number of different functions within the class. This shows how the use of interfaces can be useful in creating a framework for the code.

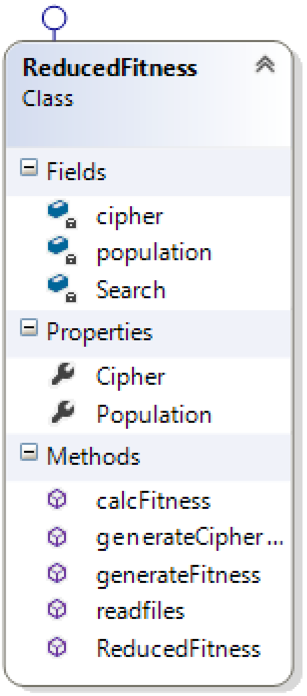
## 4.9. Fitness

The most important part of the functionality of the genetic algorithm, from the analysis I needed to complete some sort of analysis of n-gram statistics. The most extensive statistics I could find were those derived from the Google Web Trillion Word Corpus and provided by Peter Norvig[23. Attempts will be made to use both the tri-grams and the bi-grams provided to assess fitness for the algorithm. An example of some of the values from the provided bi-gram file are provided in Appendix C.

Both of the files contain an n-gram and the count of the amount of times this n-gram is found as shown in table 6.

Table 6 - Example Novig n-grams

| | |
|---|---|
| in | 134812613554 |
| the | 82103550112 |



ReducedFitness
Class

Fields
- cipher
- population
- Search

Properties
- Cipher
- Population

Methods
- calcFitness
- generateCipher...
- generateFitness
- readfiles
- ReducedFitness

Figure 30 - Example of fitness design

Obviously as the values collected for the n-grams are far too large to be used directly to calculate a fitness value, the values needed to be reduced to a more reasonable size. One way of doing this would be by normalising the values in the file. Considering that Table 6 shows those n-grams which were found the most often, normalising the data would reduce their values down to fitness values down to 1 and those other values in the files to values between 0 and 1.

As it was proposed to make a comparison of the fitness calculation types, to find the most efficient to attempt the zodiac ciphers, the implementation of the fitness algorithm will take into account both the files, or only one or the other. This would provide 3 different types of fitness to assess.

In total, the files contained 676 different bi-grams and 17576 tri-grams. Assessing these files I found that a number of the n-grams provided did not seem to be recognisable from the general use of the English language, and therefore I decided that a 4th type of fitness would be useful. By reducing the files to more recognisable n-grams, maybe the fitness functionality would be more useful.

As a final attempt to improve the fitness of the Zodiac ciphers, I will use a number of examples of words which stood out from the Zodiac's previous correspondence . These words will simply be treated as n-gram values. However they will have much higher fitness values, in an attempt to keep them in the population. These words are:

- Slaves
- Kill
- Killing
- Paradice

The values are the more prominent words from the Z408 cipher and are kept in the form they are written in that cipher, though paradise is actually spelled wrong, that is how it appears in the Z408 cipher.

To actually implement the fitness function the process would involve creating a plain text example of the cipher being assessed for each of the chromosomes in the population. The plain text will then be searched for any of the provided n-gram values. If an n-gram value is found then the associated fitness value will be added to the fitness value of the chromosome being assessed.

As all of the n-gram values will be read from a file, some sort of data structure will be required to store the n-gram string values and their associated fitness values. This will be another implemented data structure, which is shown in Figure 31. An ArrayList of these N-gram data structures can be used to assess the plain text.

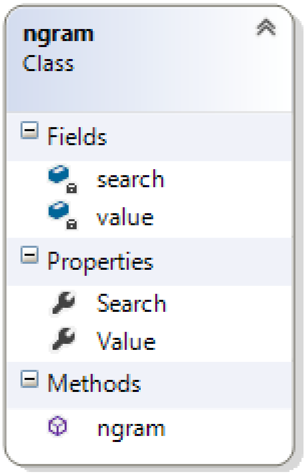


Figure 31 - N-gram design

## 4.10. Genetics

The obvious structural design for the genetic operation is to control the population from a central class, and perform mutation and crossover in separate classes, so this is how the system has been designed. The genetics class, an example shown in Figure 32, will control the population as well as the mutation and crossover rates. This will require two genetics control classes, as the mutation functionality must differ due to the finite set which makes up the chromosomes for this example.

As the genetics class will control the mutation and crossover rates, it is in the genetics class where the probability of a chromosome being part of crossover, or mutation must be calculated. Due to the fact that C# as a language does not have the functionality required to create floating point random numbers a different solution must be provided to calculate the probabilities. The solution settled upon was that as I was going to be using probabilities at a single decimal point level, a random number between 0 and 10 will be created, which would then be multiplied by 0.1 to create a decimal value which could be compared with the existing probability values. Thus any crossover or mutation rate value must be between 0 and 1, and only be of 1 decimal place.

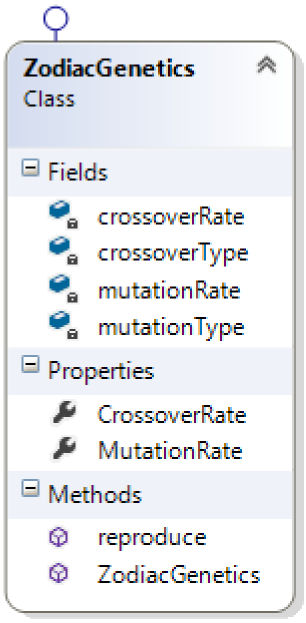


Figure 32 - Example of genetics design

The genetics class will also control which types of mutation and crossover are performed. Rather than confirm this within the Population Management class, the population management confirms which genetics class to call (Zodiac or substitution) and will pass it the information about which mutation and crossover to perform. This means that the user will only have to select which of the operators they wish to perform; the application will differentiate between the different choices and perform the correct one.

## 4.11. Mutation

Mutation is the simpler of the 2 genetic operators. However there will have to be two separate implementations. Due to the finite nature of the substitution cipher chromosome, the design must take into account the fact that random values cannot simply be placed into the chromosome. This means a substitution and a zodiac version for all mutation options must be designed. Those options are mutating a single random allele within a chosen chromosome and to assess every allele within every chromosome for mutation.

The problem of finite set mutation is a straight forward one to overcome. The substitution mutation must swap the allele chosen for mutation and replace it with another randomly chosen allele within the chromosome. This removes the possibility of any repeat alleles appearing in the population. On the other hand, Zodiac mutation must mutate an allele into any randomly chosen Latin alphabet value, as there is no knowing how many different Zodiac cipher characters will refer to which Latin alphabet characters.

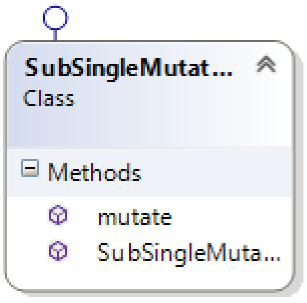


Figure 33 - An example of mutation design

Taking into account the differences between the ciphers, the mutation operators can function in much the same way. The first, by mutating one bit, a random value between 0 and the size of the chromosome will be created, and the allele in that position of the chromosome chosen in the genetics class will be mutated.

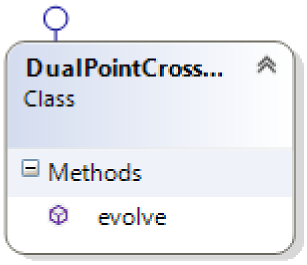
The second will be more complicated. As every chromosome must be analysed, the calculation of probability will be moved from the genetics class to the mutation class. To accomplish this, the mutation rate in the genetics class can be set to 1, passing all the chromosomes to the multi mutation class. As every allele is now being assessed for mutation, the effective mutation rate will be multiplied by 0.1. However, the simplest way to assess a two decimal place mutation rate seems to be to remove the decimal place, thereby creating an integer, a mutation rate of 0.1 becomes 1. By then creating a random number between 0 and 100 an effective probability of 0.01 is then created. Using this different way of creating probability, all alleles are then assessed, and those chosen are mutated by either the substitution cipher method or the Zodiac cipher method, depending on which is being analysed.

### 4.12. Crossover

The two options for crossover will be single point and dual point will be implemented in the traditional fashion.

For single point crossover in the case of an odd number, the crossover position will be fixed at half way through the chromosome, or at the integer rounded value of the chromosome length divided by two. The single crossover class will then create two new chromosomes. One from the first half of the first chromosome and the second half of the second chromosome, a second from the first half of the second chromosome and the second half of the first chromosome. These two new chromosomes will be passed back to the genetics class and will replace their parents in the new populations.

For dual point crossover, the implementation will be much the same again. The two crossover points will be fixed at one quarter of the chromosome and three quarters of the chromosome. This will again create two new chromosomes which will replace their parents in the new population.



The Crossover operators will not be written in a problem specific way.

Figure 34 - Crossover design

### 4.13. Selection

There are 3 different selection operators; 2 are extremely similar, roulette wheel and Stochastic universal, and the third, tournament is quite different. All however are well known selection operators in the genetic algorithm field, and will all be able to function in a non-problem-specific way.

Roulette wheel selection will function by taking the whole population and summing the fitness values for the whole population. The algorithm can then create a series of random values, within the range of 0 and this maximum fitness. By looping through the population, again summing the fitness values until they exceed this random value, can be found the chromosome to which this random roulette values refers to. This process will then be repeated until the algorithm has created a new population of the same size as the one passed into the algorithm. The fitness values being used will be floating point values between 0 and 1. However, as C# does not have a floating point random value creation option, this may unfortunately reduce the effectiveness of this selection operator, but it will function.

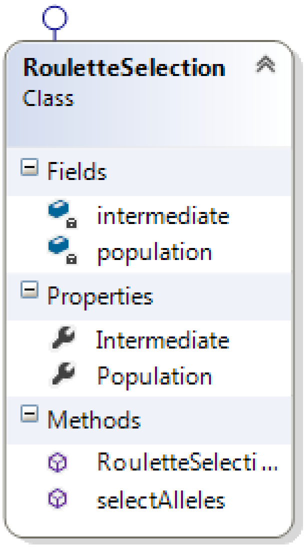


Figure 35 - Selection design example

Stochastic universal sampling will function in a very similar way to Roulette wheel selection. The same algorithm will be used to sum the total population fitness value, and the same algorithm will be used to sum fitness until it exceeds the chosen value. Where stochastic universal sampling will differ is in picking values. The process will this time only use one random value, created by dividing the total population fitness value by the amount of chromosomes that are needed for the new population, thus creating an interval value which would divide the population up equally. By then creating a random value between 0 and this interval value, the value used in selection will be created. The algorithm will then pick the chromosomes for which the summed fitness exceeds that of the summed interval value. This gives a more level sampling of the old population in the new population.

Tournament selection operates in a completely different way, though it will be more straight forward. The process will be that there will be a random selection of a certain percentage of the original population, from which the population is being selected. This set of selected chromosomes will then be looped through to find the highest fitness in the selection. The chromosome with the highest fitness will then be placed in the new population. The process will be repeated until full population is created. The higher the size of the tournament, the more focus there is on the chromosomes with the highest fitness. A high tournament size will be picked, probably 20% of the population, to focus more on the fittest solutions.

### 4.14. Termination

Termination is the most straight forward section to design. With the choice to only implement a time based, it is a very simple algorithm. With the GUI allowing the input of a time limit, I decided that the most logical increment of time to use was minutes. Seconds was far too small and hours would be programmable with larger lengths of minutes. All that is required here is to take the current time at the start of the algorithm, which can then be compared with the time taken when running the doesTerminate method. This can then return a Boolean value depending on the comparison of times.

### 4.15. Ciphers

As the ciphers are modelled numerically for investigation, this seems like the easier way to model the ciphers for the application. Rather than use a matrix in the same form as the cipher written by the zodiac, I will model them by creating a data structure which will contain the locations of each character within the cipher, as shown in Table 7. Table 7 only shows a selection of the character locations, not the complete 63 characters of the Z340 cipher.

Table 7 - Z340 character location example

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 50 | 177 | 66 | 99 | 19 | 57 | 56 | 60 | 80 | 84 | 77 | 242 | 47 | 184 | 43 | 68 | 76 |
| 301 | 248 | 88 | 249 | 53 | 91 | 89 | 115 | 298 | 97 | 119 | 278 | 118 | 197 | 114 | 100 | 95 |
| 327 |  | 144 | 324 | 54 | 156 | 160 | 158 |  | 309 | 178 |  | 135 | 275 | 121 | 110 | 137 |
|  |  | 173 |  | 62 | 191 | 226 | 234 |  | 330 | 194 |  | 277 | 302 | 223 | 122 | 310 |
|  |  | 192 |  | 86 | 222 | 266 | 250 |  |  | 213 |  | 289 |  | 271 | 145 |  |
|  |  | 221 |  | 164 | 284 |  |  |  |  | 256 |  |  |  |  | 188 |  |
|  |  | 273 |  | 182 |  |  |  |  |  | 287 |  |  |  |  | 229 |  |
|  |  |  |  | 265 |  |  |  |  |  | 307 |  |  |  |  | 272 |  |
|  |  |  |  | 306 |  |  |  |  |  | 337 |  |  |  |  |  |  |
|  |  |  |  | 329 |  |  |  |  |  |  |  |  |  |  |  |  |

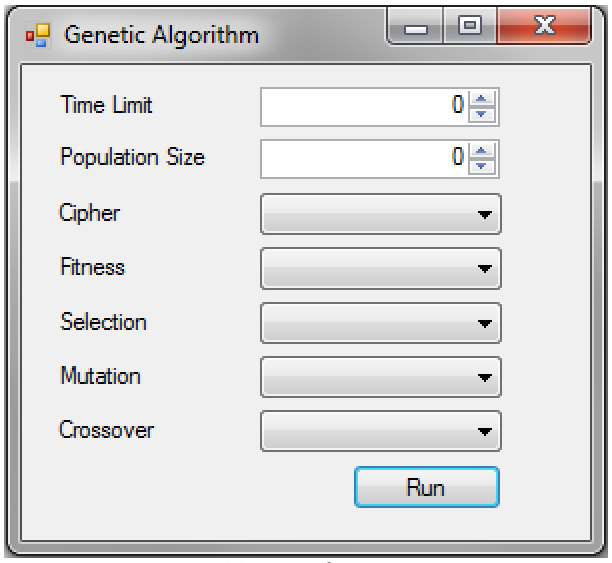
By using this form, it eases the creation of a plain text from the chromosomes which have already been designed. Using a data structure like this, each character can be placed in its correct position in the plain text one at a time, looping through the whole data structure to fill the plain text.

Full UML class diagrams for all parts of the program can be found in Appendix D.

# 5 Implementation

## 5.1. Application

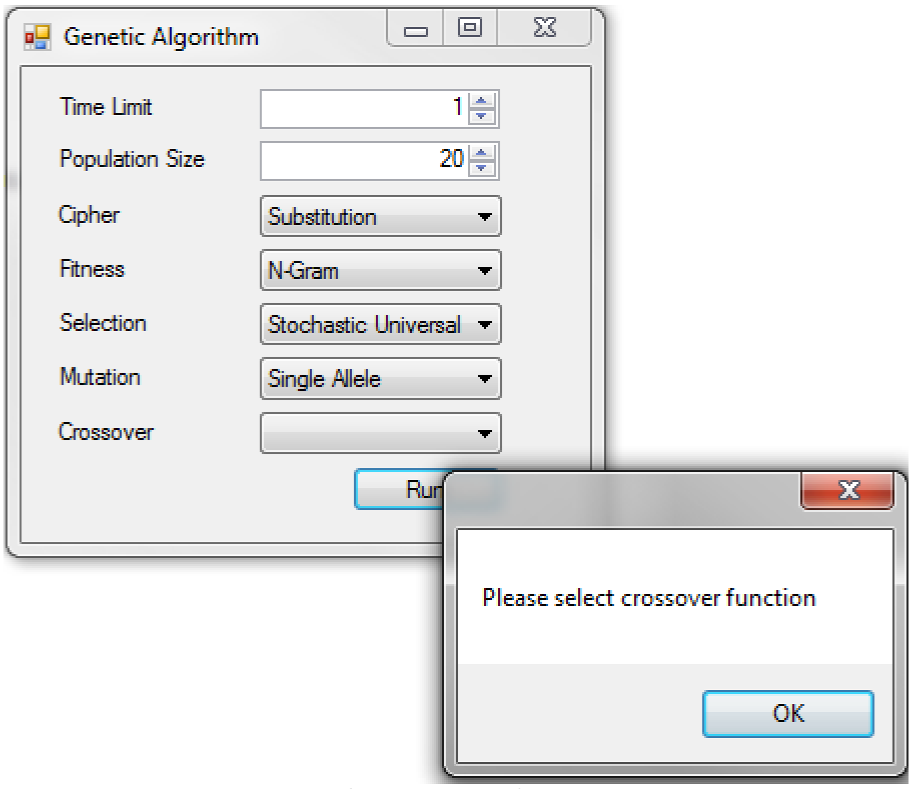
The application is simple and easy to use, as it is so simple, the final GUI is laid out as shown in Figure 36.


Genetic Algorithm

Time Limit: 0
Population Size: 0
Cipher:
Fitness:
Selection:
Mutation:
Crossover:

Figure 36 - Final GUI

Though it is only a research tool, there is some verification in to stop fatal errors, in the cases where the user may have not selected an option, shown in Figure 37. In terms of number verification, the c# number input box only allows number input, therefore the implementation did not have to worry about this creating problems within the program.

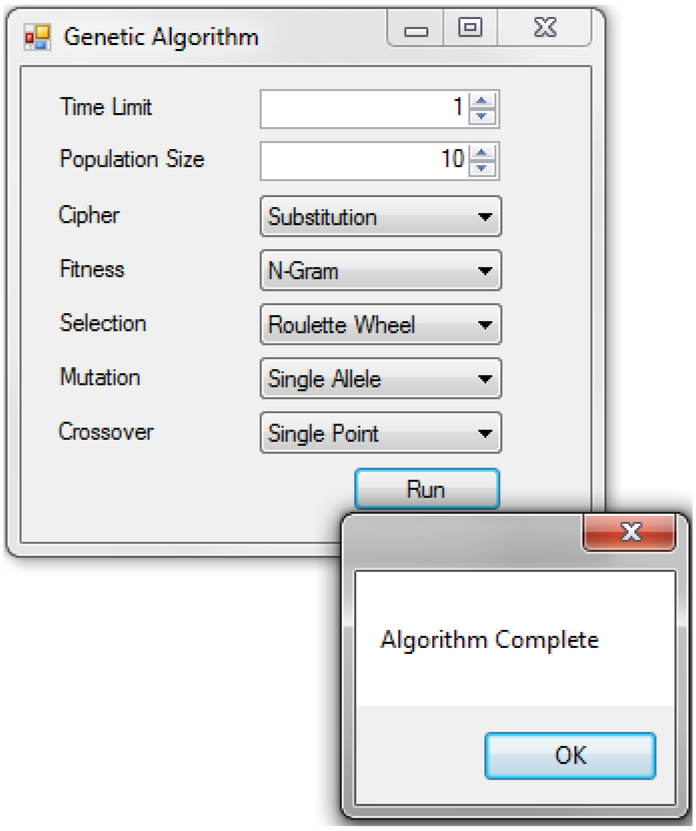

Genetic Algorithm

Time Limit: 1
Population Size: 20
Cipher: Substitution
Fitness: N-Gram
Selection: Stochastic Universal
Mutation: Single Allele
Crossover:

Please select crossover function

Figure 37 - GUI selection verification

Finally there is an implemented message box to inform the user of the completed algorithm. Unfortunately it is not possible to focus on the program when the algorithm completes, as the functionality of c# does not allow this, but I felt some sort of indication was needed, this is shown in Figure 38.



Time Limit: 1
Population Size: 10
Cipher: Substitution
Fitness: N-Gram
Selection: Roulette Wheel
Mutation: Single Allele
Crossover: Single Point

Algorithm Complete

Figure 38 - Algorithm completed indicator

### 5.2. Output

The output of the genetic algorithm is placed in a text file created in the directory of the application executable file. The output takes the form of maximum fitness, mean fitness, cipher key. The final entry at the end is the fitness of the final result and the full plain text. An example of output would be:

```
32.9525288215896
12.5466684364806
smlvpcdtixhzenguawrqfbkojy
32.9525288215896
12.5638296887786
smlvpcdtixhzenguawrqfbkojy
32.9525288215896
12.5764490143126
smlvpcdtixhzenguawrqfbkojy
32.9525288215896
smlvlvpvcdvstscslixhlzmengincvelxsmesevstxuigazteheiguexesthawuintsmrvigsqaneqtsms
meatrignefchtxusmeebzentrexsastixvzahegingcnsmenrinehirzwthasefhtzmeng
```

# 6 Investigation Methods

## 6.1. Substitution

The implementation of the substitution cipher is to use a cipher which should be simpler to solve to complete a number of investigations into all the options offered by the implemented genetic algorithm. From these results it will hopefully be more clear which are the best methods offered by the genetic algorithm which can then be used to analyse the Zodiac ciphers.

The reason for the use of a separate cipher, is that there are a huge range of variable which can be altered and have an effect on the cipher output. In this case, I wish to look at the differences in a number of the operators which have been implemented, these being

| | | |
|---|---|---|
| • Fitness function | - | Bi-gram, tri-gram, N-gram, reduced |
| • Selection | - | Roulette, stochastic, tournament |
| • Mutation | - | Single Allele, Multi Allele |
| • Crossover | - | Single Point, Dual Point |

In order to look at these variables, there are a number of other variable which could have an effect which must be fixed, in order to make experimentation fair, these are:

| | | |
|---|---|---|
| • Time | - | 30 Mins. |
| • Population size | - | 100 |
| • Crossover rate | - | 0.6 |
| • Single Mutation rate | - | 0.1 |
| • Multi Mutation rate | - | 0.05 |

Even with these variables fixed this still gives a total number of 48 different ways in which the genetic algorithm can be run. It would take too long to run all of these against the Zodiac ciphers.

To investigate the substitution cipher I needed a substitution cipher of long enough length to be solvable, but not too long or complicated to be too hard to be comparable to the Zodiac ciphers. The cipher I decided on was this:

*"This is a substitution cipher for use in the testing of a piece of Genetic Algorithm software with the aim of reducing the experimentation space for further more complicated ciphers."*

I then enciphered this text using the substitution cipher tool provided by T. Akins at rumkin.com[18]. I used the pigpen alphabet #X#X alphabet, in order to imitate the use of a different alphabet by the zodiac cipher. As well as attempting to show that with the numeric modelling technique, the alphabet used should not make any difference. In a way, using numbers is simply another, more understandable alphabet. The cipher text and numerical model can be found in Appendix E.

To work out the most efficient way of working with the genetic algorithm which has been implemented, I could work with the substitution cipher model for thousands of results. However, rather than spend what time I had working with the substitution model I wished to move onto the Zodiac ciphers more quickly. Due to this, I only created one set of results for the substitution cipher model, and made the assumption that this would provide good results for the genetic algorithm. This is a risky assumption to make, as genetic algorithms can be very unpredictable, but with limited time, I wanted to assess the Zodiac Ciphers with what I had.

## 6.2. Zodiac Ciphers

There are 2 Zodiac ciphers which can be used for investigation, the Z408 and the Z340, both of which have been modelled in the implementation of the genetic algorithm. The Z340 is the more interesting problem, however the Z408 has the added benefit of already having a known solution, allowing easy assessment of how successful the genetic algorithm has actually been.

Using the larger scale results from the investigation of the substitution cipher, the three most successful sets of options will be used to investigate each of the zodiac ciphers. Investigating the Z408 cipher with the aim of proving that the algorithm can attempt to solve this sort of homophonic substitution cipher. Attempting the Z340 cipher simply to see if any further information about the ciphers can be gleaned about the cipher itself, there is no real confidence in whether the algorithm will find anything new or not.

By only using 3 different option selections, the algorithm can be run more than once to find more information. As well as running the algorithm multiple times for each of the options. It may also be useful to alter the time and population size for the algorithm, by increasing the population size, it improves the chances of good initial chromosomes, and by increasing the time it gives the algorithm more possible generations to complete. Initially the algorithm will run one increased time and population trial, and depending on the results of this, may repeat the shorter investigations, of the same length as those used for the substitution assessment.

As the prior reading concluded that a population of 300 was the most efficient[15], it may produce positive results if the population size is increased to this value. As the population size is increasing, it will be necessary to increase the running time of the algorithm to keep approximately the same number of generations, therefore running time will be increased to 60 minutes.

Changing variables will be:

| | | |
|---|---|---|
| • Time | - | 60 Minutes. |
| • population size | - | 300 |

Again the fixed variables will be:

| | | |
|---|---|---|
| • Crossover rate | - | 0.6 |
| • Single Mutation rate | - | 0.1 |
| • Multi Mutation rate | - | 0.05 |

# 7 Investigation Results

## 7.1. Substitution

Table 8 - Substitution Cipher Results

| Cipher | Time | Pop Size | Fitness | Selection | Mutation | Crossover | Max. Final Fitness | Change in Mean Fitness | No. of Generations |
|---|---|---|---|---|---|---|---|---|---|
| Sub | 30 | 100 | Bi-gram | Roulette | Multi | Dual | 32.95253 | +0.764741 | 23209 |
| Sub | 30 | 100 | Bi-gram | Roulette | Multi | Single | 31.45043 | +0.647488 | 23277 |
| Sub | 30 | 100 | Bi-gram | Roulette | Single | Dual | 28.74212 | +0.473866 | 23419 |
| Sub | 30 | 100 | Bi-gram | Roulette | Single | Single | 29.91333 | +0.580506 | 23829 |
| Sub | 30 | 100 | Bi-gram | Stochastic | Multi | Dual | 31.67773 | +0.369095 | 23842 |
| Sub | 30 | 100 | Bi-gram | Stochastic | Multi | Single | 30.48168 | -0.54108 | 22807 |
| Sub | 30 | 100 | Bi-gram | Stochastic | Single | Dual | 28.38565 | -0.09459 | 22715 |
| Sub | 30 | 100 | Bi-gram | Stochastic | Single | Single | 30.82122 | +0.415788 | 24301 |
| Sub | 30 | 100 | Bi-gram | Tournament | Multi | Dual | 28.49615 | -5.65722 | 23546 |
| Sub | 30 | 100 | Bi-gram | Tournament | Multi | Single | 29.23033 | -4.75474 | 23218 |
| Sub | 30 | 100 | Bi-gram | Tournament | Single | Dual | 28.73644 | -4.08619 | 23095 |
| Sub | 30 | 100 | Bi-gram | Tournament | Single | Single | 29.67399 | -4.78903 | 24189 |
| Sub | 30 | 100 | Tri-gram | Roulette | Multi | Dual | 3.704801 | -0.04266 | 1239 |
| Sub | 30 | 100 | Tri-gram | Roulette | Multi | Single | 4.201548 | -0.08334 | 1181 |
| Sub | 30 | 100 | Tri-gram | Roulette | Single | Dual | 3.318412 | +0.0656 | 1277 |
| Sub | 30 | 100 | Tri-gram | Roulette | Single | Single | 3.258335 | +0.00004 | 1187 |
| Sub | 30 | 100 | Tri-gram | Stochastic | Multi | Dual | 2.372034 | +0.15794 | 1237 |
| Sub | 30 | 100 | Tri-gram | Stochastic | Multi | Single | 2.114218 | +0.01055 | 1178 |
| Sub | 30 | 100 | Tri-gram | Stochastic | Single | Dual | 3.079751 | -0.00308 | 1277 |
| Sub | 30 | 100 | Tri-gram | Stochastic | Single | Single | 2.944127 | +0.023606 | 1184 |
| Sub | 30 | 100 | Tri-gram | Tournament | Multi | Dual | 3.921881 | -0.23259 | 1235 |
| Sub | 30 | 100 | Tri-gram | Tournament | Multi | Single | 3.108056 | -0.2948 | 1182 |
| Sub | 30 | 100 | Tri-gram | Tournament | Single | Dual | 3.051274 | -0.17401 | 1277 |
| Sub | 30 | 100 | Tri-gram | Tournament | Single | Single | 4.836386 | -0.16065 | 1182 |
| **Sub** | **30** | **100** | **N-gram** | **Roulette** | **Multi** | **Dual** | **39.61359** | **+0.427915** | **1105** |
| Sub | 30 | 100 | N-gram | Roulette | Multi | Single | 30.82973 | -0.96643 | 1242 |
| Sub | 30 | 100 | N-gram | Roulette | Single | Dual | 29.7487 | 0.072991 | 1094 |
| Sub | 30 | 100 | N-gram | Roulette | Single | Single | 27.76916 | -0.94625 | 1447 |
| Sub | 30 | 100 | N-gram | Stochastic | Multi | Dual | 32.74535 | +1.083171 | 1103 |
| Sub | 30 | 100 | N-gram | Stochastic | Multi | Single | 30.16287 | +0.84476 | 1293 |
| Sub | 30 | 100 | N-gram | Stochastic | Single | Dual | 30.87677 | -0.07297 | 1091 |
| Sub | 30 | 100 | N-gram | Stochastic | Single | Single | 24.64982 | -0.09789 | 1323 |
| Sub | 30 | 100 | N-gram | Tournament | Multi | Dual | 26.39399 | -2.06303 | 1105 |
| Sub | 30 | 100 | N-gram | Tournament | Multi | Single | 24.42757 | -1.11999 | 1203 |
| Sub | 30 | 100 | N-gram | Tournament | Single | Dual | 28.7542 | -1.67207 | 1096 |
| Sub | 30 | 100 | N-gram | Tournament | Single | Single | 23.83168 | +1.12613 | 1233 |

Table 9 - Substitution Cipher Results continued

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Sub | 30 | 100 | Reduced | Roulette | Multi | Dual | 34.78968 | -0.32207 | 36532 |
| **Sub** | **30** | **100** | **Reduced** | **Roulette** | **Multi** | **Single** | **36.81026** | **+1.499492** | **34723** |
| Sub | 30 | 100 | Reduced | Roulette | Single | Dual | 31.587 | +1.126452 | 36010 |
| Sub | 30 | 100 | Reduced | Roulette | Single | Single | 31.09353 | +0.586218 | 37475 |
| **Sub** | **30** | **100** | **Reduced** | **Stochastic** | **Multi** | **Dual** | **36.71944** | **+0.637962** | **35908** |
| Sub | 30 | 100 | Reduced | Stochastic | Multi | Single | 35.63998 | +1.005825 | 35662 |
| Sub | 30 | 100 | Reduced | Stochastic | Single | Dual | 32.02201 | +0.048737 | 36726 |
| Sub | 30 | 100 | Reduced | Stochastic | Single | Single | 30.75373 | +0.247218 | 37320 |
| Sub | 30 | 100 | Reduced | Tournament | Multi | Dual | 27.09435 | -4.86755 | 37544 |
| Sub | 30 | 100 | Reduced | Tournament | Multi | Single | 26.88056 | -6.4987 | 36790 |
| Sub | 30 | 100 | Reduced | Tournament | Single | Dual | 27.82926 | -5.58989 | 37231 |
| Sub | 30 | 100 | Reduced | Tournament | Single | Single | 31.28448 | -5.45016 | 37764 |

From these results, the 3 best combinations of features which give the best results are required. However the need arises to compare all of the results on the same level. Due to the normalisation of the data, the fitness values of the tri-gram fitness has been calculated as much lower than that including the bi-gram data. To further compare the data involving just tri-grams, there would need to be some way of increasing the value of the data to a comparable level. This could be as simple as multiplying the values by 10, which creates values of similar size, however this could also be a mistake and could skew the actual data.

Thankfully, there are not only fitness values to consider, there is also consider the change in mean fitness across the entire population. As the genetic algorithm is mean to increase overall fitness, a decrease in fitness over the total population is quite a bad sign overall, therefore from this data, it would be best not to consider any data which has a negative population fitness change. This reduces the set of operators which could be passed on to assess the Zodiac ciphers, as well as removing the highest values from the tri-gram fitness assessments.

This leaves three choices, based on the assessments which have the highest fitness, and have not resulted in degradation of the overall population fitness. These are shown in bold in Tables 8 and 9 and are as follows:

1. N-gram fitness with roulette wheel selection, multiple allele mutation and dual point crossover.
2. Reduced Fitness with roulette wheel selection, multiple allele mutation and single point crossover.
3. Reduced fitness with stochastic universal sampling, multiple allele mutation and dual point crossover

To assess how well they have attempted the substitution cipher the plain texts created by the output of these attempts are as follows

1. tiyryrvrworthtwtyzknyliesuzswreyktieterthkpzumlhenezupekethnmgpzshtifrzutjmsejhtitiemhf zusedwnhkptieecleshfektmthzkrlmneuzsuwstiesfzsenzflghnmtednhliesu

2. thsfsfufaqftctatsdmnsrhelxdlafesmtheteftcmjdxircenedxjemetcnibjdlcthvfdxtkilekcththeicvdxl eoancmjtheeprelcvemtitcdmfrinexdlxalthelvdlendvrbcniteoncrhelx

3. thucucbcfoctjtftusvpuxherqsrfceuvthetectjvisqaxjepesqievetjpadisrjthwcsqtlareljththeajwsqr enfpjvitheeyxerjwevtatjsvcxapeqsrqfrtherwsrepswxdjpatenpjxherq

These can be compared to the plain text which was enciphered, which was:

*This is a substitution cipher for use in the testing of a piece of Genetic Algorithm software with the aim of reducing the experimentation space for further more complicated ciphers*

Though these attempts are not managing to find the actual solution to the cipher, they are moving in the right direction. It may be that the cipher would simply require more time to move further towards the correct solution. But as these are the most successful attempts made, these are the three settings selected to move on and attempt the Zodiac ciphers.

### 7.2. Initial Zodiac results

Table 10 and Table 11 show the initial results from analysing the Z408 and Z340 ciphers with the three genetic algorithm types. It can be seen that the unfortunately all of the collected results for both the Z408 and the Z340 are demonstrating a downward trend in mean fitness change. This is a very bad sign in terms of the success of the algorithms. If we also analyse Figure 39, a graph of the highest generated fitness for either the Z408 and the Z408 we see that there is very little increase in the best chromosome from initial population to algorithm completion.

Table 10 - Initial Z408 results

| Cipher | Time | Pop Size | Fitness | Selection | Mutation | Crossover | Max. Final Fitness | Change in Mean Fitness | No. of Generations |
|---|---|---|---|---|---|---|---|---|---|
| Z408 | 30 | 100 | N-gram | Roulette | Multi | Dual | 99.72483 | -44.8938 | 432 |
| Z408 | 30 | 100 | Reduced | Roulette | Multi | Single | 92.40718 | -49.2351 | 13623 |
| Z408 | 30 | 100 | Reduced | Stochastic | Multi | Dual | 101.4459 | -52.7324 | 14156 |
| Z408 | 60 | 300 | N-gram | Roulette | Multi | Dual | 106.4436 | -39.4532 | 266 |
| Z408 | 60 | 300 | Reduced | Roulette | Multi | Single | 101.8384 | -44.7984 | 8245 |
| Z408 | 60 | 300 | Reduced | Stochastic | Multi | Dual | 100.4283 | -55.1183 | 9138 |

Table 11 - Initial Z340 results

| Cipher | Time | Pop Size | Fitness | Selection | Mutation | Crossover | Max. Final Fitness | Change in Mean Fitness | No. of Generations |
|---|---|---|---|---|---|---|---|---|---|
| Z340 | 30 | 100 | N-gram | Roulette | Multi | Dual | 88.95457 | -46.0523 | 502 |
| Z340 | 30 | 100 | Reduced | Roulette | Multi | Single | 85.19211 | -28.459 | 14731 |
| Z340 | 30 | 100 | Reduced | Stochastic | Multi | Dual | 83.99485 | -44.7712 | 15790 |
| Z340 | 60 | 300 | N-gram | Roulette | Multi | Dual | 90.62201 | -38.8652 | 341 |
| Z340 | 60 | 300 | Reduced | Roulette | Multi | Single | 91.12437 | -39.8014 | 10349 |
| Z340 | 60 | 300 | Reduced | Stochastic | Multi | Dual | 86.00219 | -45.5751 | 11223 |

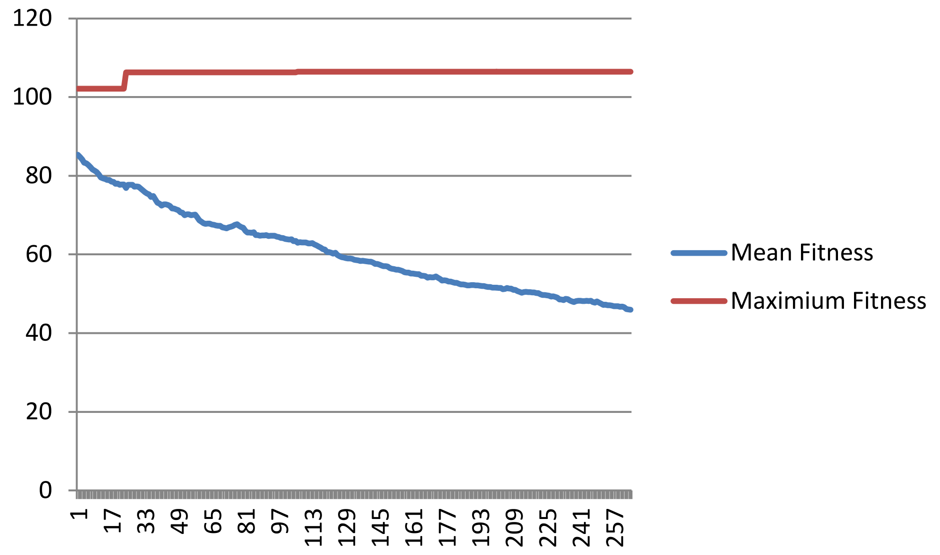


Figure 39 - Graph of maximum fitness and mean fitness for an algorithm of Time 60, Population 600, N-gram fitness, Roulette wheel selection, Multi allele mutation and Dual point crossover analysing the Z408 cipher.

If we look at the plain text completed from the algorithm displayed in Figure 39, shown below, we see that there is little or no resemblance to the actual plain text of the Z408 cipher.

*eoesmsohorhttcrthdewoisteeretufosotcsioiraondeeshagcisekhehthoortiodriatmcoelctredoastwoiheui georaamcitedfsdihaoih**neck**haosekhtrodrtoittrimuocigdofaaateekhehtwnteleiodorrtmichediawdagdit eriehtnnseloostrce**heat**etodoageedurtieifcraeaatiehgmiemocrhekhedlwendheitceirooerhmiooitergci dseoomrhekhecorodontkaiwtehehohnatoiwnosonicodemoisudnnsheokadnrftoremrhhosraftonookke oaehtretoiimtcndonaciweheedeedreedieodrtgtoir*

One thing which can be considered is that for these instances of the genetic algorithm, those which ran for longer and used a larger population consistently created results which have higher fitness values, and less bad changes in mean population fitness. The differences in mean fitness could be due to differing number of generations, but the difference in fitness could mean that the longer algorithm is the better option. Following this logic, further investigation was carried out with the longer timeline and larger population.

## 7.3. Further Z340 results

Further investigation into the Z340 cipher with these Genetic algorithm operators produced further results, the most effective of which are shown in Table 12. Full Z340 cipher results are shown in Appendix E. The cipher, time and population size were all fixed for these attempts at the Z340 cipher, at 60 minutes and 300 chromosomes, so these values have been removed from the table.

Table 12 shows values for the 4 highest recorded final fitness values, and the highest value for change of final fitness throughout the algorithm.

Table 12 - Z340 investigation results

| Fitness | Selection | Mutation | Crossover | Max. Final Fitness | Change in Final Fitness | Change in Mean Fitness | No. of Generations |
|---|---|---|---|---|---|---|---|
| N-gram | Roulette | Multi | Dual | 94.78381 | 0 | -59.9092 | 338 |
| N-gram | Roulette | Multi | Dual | 92.20771 | 0 | -54.1404 | 338 |
| Reduced | Roulette | Multi | Single | 91.12437 | 0 | -39.8014 | 10349 |
| Reduced | Stochastic | Multi | Dual | 90.40165 | 0 | -69.7312 | 10803 |
| Reduced | Roulette | Multi | Single | 89.07682 | 1.858945 | -58.2668 | 9961 |

It is clear that the reduction in the overall fitness of the fitness of the population is having an extremely adverse effect on the ability of the genetic algorithm find a suitable solution to the Zodiac ciphers. Looking at both Figure 39 and Figure 40 it is clear that any increase in maximum fitness occurs near the start of the cycle of generations, as the mean fitness initially falls. As the populations overall fitness falls, it is impossible to increase the best solution further.

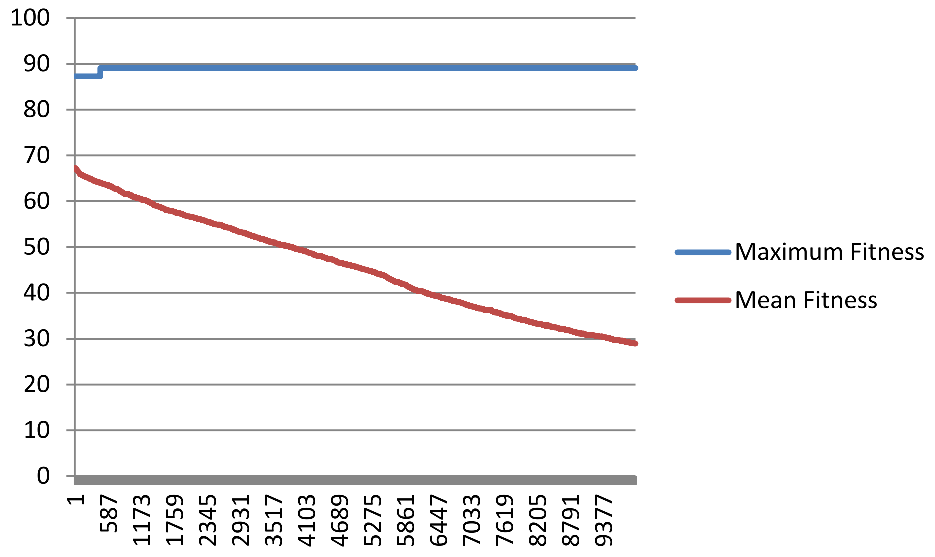


Figure 40 - Graph of maximum fitness and mean fitness for an algorithm of Time 60, Population 600, reduced set fitness, roulette wheel selection, multi allele mutation and single point crossover analysing the Z340 cipher

Table 12 also shows that the maximum fitness results often never increase at all, the highest fitness values created by the algorithm are all those which have been created in the initial population, and due to the use of elitism, this fittest chromosome has not been allowed to leave the population.

We can also assess the plain text created by the chromosome with the highest fitness, 94.78.

*uauwles**net**svyconrdlstelahacaimohhntortesedocenyleueklloseeondlassuhniieselorsphesintrleushe arortthwnaelpshthtnetfonieyseoneholasidaesayirmesbsdunhttogsnhreeenssesalomhtoresurplasaatl bcehanmteuesopcneisithesikolsdsphtteteueohdshpnnbitnnlssotfvornitawnedprsstidet**heal**sihetonut csyvstmshectsnyssnctecteaeucrenlshtrcmoltiaatorttweaudltkedaegshts*

Again the decryption of the cipher text does not reveal any useful data, with the longest recognisable words being those in bold, and only 4 characters long at longest.

# 8 Conclusions

## 8.1. Substitution ciphers

Though the algorithm was unable to correctly complete the task of solving the substitution cipher, this was not the main aim of this research, however it is possible to make some conclusions referring to the algorithms attempts on the substitution cipher.

Firstly and most obviously, the implementation of the tournament selection was ineffective in selecting a good population for the continuation of the algorithm. The selection method consistently returned a high overall decrease in the fitness of a population, only once generating a positive value in this regard. This would probably be because the tournament size chosen for this assessment was too high. The focus on attempting to select the highest fitness chromosomes in the population could have reduced variation too quickly in the population and stagnated it, as there was not a high enough mutation rate to keep variation high enough.

Secondly it can be seen that combining the fitness values of the provided bi-grams and tri-grams in this way was ineffective. There needed to be some way of combining the values which gave more weight to the tri-grams in this calculation, however as can be seen from the tri-gram only fitness, the values were much smaller. Tri-grams could have been much more important in the cipher cryptanalysis, as they are larger sections of words and cover more characters in a cipher. There may be some way of normalising all of the n-grams together, or there may be some way of increasing the tri-gram values, which may have improved fitness functionality.

## 8.2. Zodiac ciphers

It can be concluded that this investigation was unsuccessful in its overall aim to attempt to solve the Zodiac ciphers, an intelligible plain text could not be found for either the Z408 or the Z340. This could have been due to any number of variables however the most likely are either the fitness function, or how the fitness values interacted with the selection operation. The separated bi- and tri-grams created an unfortunate situation where tri-gram values generated much smaller fitness values than bi-grams, despite technically being more important to the fitness, due to their longer length and lower frequencies. On the other side of the same coin, the higher bi-gram values may have skewed results by giving high values to situations where the bi-grams were out of place.

Another point, could be that the files used to provide the n-gram values contained a huge number of n-gram examples, a large number of which were not recognisable in terms of normal English text. The reduced set fitness function was an attempt to only take more common n-grams, but this seems to have made little difference in functionality. It might be necessary to reduce the set even further to a smaller set of recognisable n-gram values.

In terms of the fitness function, it may be that as the changes in the values of fitness were so small, between 0 and 1 to quite large decimal place values, there was not enough variation in the population fitness to differentiate between good solutions and bad ones. The operation of a genetic algorithm depends the selection pressure of the selection operator, the degree to which better solutions are emphasised by the selection operation. If the selection operator cannot differentiate between good solutions and bad solutions because the fitness values are too similar, the algorithm is not going to function properly.

One area that this investigation did succeed in was creating a framework for a genetic algorithm to be built upon. Though this first permutation has not been completely successful, it has had some success, which could be built upon to further investigate the framework, or genetic algorithms in cryptanalysis.

## 8.3. Future Work

As the genetic algorithm did have some success in improving populations when attempting the substitution cipher, it could be possible to further develop this genetic algorithm model to continue research in this area. The overall functionality seems sound, and with further development of the fitness functionality better results may be forthcoming.

However, overall, to develop further in this area, more research will be required on both the Zodiac ciphers, and in the general area of using genetic algorithms for cryptanalysis. As concluded by B. Delman, it may be that genetic algorithms are not a useful tool in this area and focus should remain on traditional cryptanalysis tools[4].

# 9 Critical Evaluation

## 9.1. Overview

Overall, most of the design and implementation of what has been presented in this project has gone well, there have been some points where I have had to change things I was not expecting, and there are a couple of bad design choices. However even though the final result of the project was not what I was hoping, I feel that overall it has gone well. In terms of a culmination of my time at university, the use of an object oriented language was very familiar, and to expand my knowledge in one specific area of artificial intelligence was a very satisfactory way to complete my degree, after taking all of the artificial intelligence modules. Though I am enrolled on the overarching computer science degree scheme, over the years, artificial intelligence became the focus of most of my module choices.

## 9.2. Technology & Tools

Most of the technology I was using was familiar to me. Developing in C# with Visual Studio was comfortable, as it had become my standard setup while working during my industrial year. The only other technological tool I used was GitHub, and though I did not require much functionality provided by a version control system, it was nice to know that my work was stored in the cloud. The functionality of GitHub, though not specifically familiar with the technology, was similar enough to previous version control systems for me to pick up quickly.

One tool I did  not use, which would have been useful, was the use of a diary. Though I kept all my documents, and kept the locations of all the papers I read, there is no way I will have remembered everything that took place during the development process. I did attempt to start taking a diary, however, while also trying to follow a new process, feature driven development, and trying to re-acquaint myself with the ideas of genetic algorithms, this is one idea which fell by the wayside quite quickly.

## 9.3. Design

On the design front, I believe that Feature Driven Development was a good choice for this project. The design process forced me to think in a modular way, and created a good overview and interface design from the beginning. It also allowed me to work on one part of the design at a time, and due to the amount of research I needed to do for each section this helped keep all of the knowledge segmented somewhat, rather than overwhelming me with the specifics of a subject for which I had only a small knowledge of. However, following the design phase, feature driven development was not quite such a good choice, or maybe I just implemented the process badly.

## 9.4. Process

Following the design, I needed to do some spike work, to investigate the subject further and to check my design decisions, however this spike work was not completed in a test driven development (TDD) fashion. As I was new to the TDD process, I wanted to simply improve my knowledge of the area. This spike work however slowly transformed into more of an actual solution, as the mid project demo approached, and I had need of a working prototype to demonstrated. Following the mid project demo, this code then provided the basis for my actual development, and as the core was not developed in a TDD way, I decided simply to abandon this part of the feature driven development process. This in turn meant that the development code was probably not tested as thoroughly as it should have been. In hindsight, as I was new and unfamiliar with TDD I should probably have simply completed all of written code in this way from the start. It would have meant I became accustomed to using the process, and the method of coding probably would have improved the quality of the research.

## 9.5. Implementation

In terms of the implementation of the project, I think that the idea of comparing differing genetic algorithm methods to find the most suitable was a good idea in theory, and with further development would be a useful method of attempting to analyse ciphers. I also believe that the implementation of the genetic algorithm operators that I have implemented was good, and would be useful in further investigation, as the operators are already implemented and ready. However, in light of the unsuccessful nature of the research, and the likely reasons discussed in the conclusions, it may have been a better idea to reduce the amount of external factors, and focus more on the problem specific operators. For example, with more investigation into fitness functions, I could have found a more useful fitness function, and compared that with the currently implemented method. Rather than just using a number of variations on the same implementation which is what is currently shown in this research.

## 9.6. Previous Experience

This year, having returned from a year in industry, it could be said that I picked up some bad development habits while I was away from university. The use of manual testing over automated testing especially seems to have taken hold and has been used in this project, which somewhat overrode my decision to use test driven development as well as simply being a bad way of developing software.

The experience of working in software development environment also seems to have had an impact on my approach to work. I should have spent more time thinking about the project in the beginning, as I began approaching it in a very software development fashion, rather than with a research mindset. This had to change over the course of the investigation, however if this had been my mindset at the beginning I could have changed my development pattern. Rather than one iteration of design, implement, assess, I could have had more than one iteration of assessing the substitution and Zodiac ciphers. This would have given me the opportunity to solve some of the problems encountered during the investigation phase of the project.

## 10 Bibliography

[1] J. M. Carrol and S. Martin, “The Automated Cryptanalysis of Substitution Ciphers,” *Cryptologia,* vol. 10, no. 4, pp. 193-209, 1986.

A very old paper, I did not use a lot of information from this article. Some of the overall design was useful, however due to the fact that the authors used prolog as their language of choice, there was not a large amount of synergy with the object oriented fashion in which I wanted to write my genetic algorithm example.

[2] R. Spillman, M. Janssen, B. Nelson and M. Kepner, “Use of A Genetic Algorithm in the Cryptanalysis of Simple Substitution Ciphers,” *Cryptologia,* vol. 17, no. 1, pp. 31-44, 1993.

A paper useful mainly at the high level of design, had some very useful investigation of the high level genetic algorithm process, and some interesting discussion of how to complete mutation and crossover while working with a finite set chromosome, for a substitution cipher. Also used some low level mathematical formulas based on n-gram frequency statistics for calculating fitness which may have been useful for further work, if the fitness function were to be refined.

[3] B. Delman, *Genetic Algorithms in Cryptography,* Rochester Institute of Technology, Rochester: Unpublished Masters Thesis, 2004.

A Useful paper which brought together a large amount of the prior work done in the area of genetic algorithm cryptanalysis. I used this paper mostly as a springboard to find other work in the area, as the author focused on repeating previous experiments to measure success. However the author did have some interesting points on whether the use of genetic algorithms in cryptanalysis is a worthwhile endeavour.

[4] D. Thẳng, “Analysis of the Zodiac 340-Cipher,” Unpublished Master's Thesis, San Jose State University, California, 2008.

As an in depth investigation into the structure of the Zodiac ciphers I used this paper extensively in my research about how to model the ciphers computationally, as well as looking into ways I could take into account the errors made by the zodiac in the ciphers. If I could explain how the mistakes were made I could possibly work around them, or alter my computational models in such a way to make the genetic algorithm succeed. The use of the hill climb algorithm in an attempt to solve a homophonic cipher was not directly related to my work, but the optimisation ideas may have been a useful in attempting methods of optimising the genetic algorithm.

[5] “Feature Driven Development Processes” [Online]. Available: http://www.nebulon.com/articles/fdd/download/fddprocessesUSLetter.pdf. [Accessed 4 February 2014].

In describing the methods of using FDD this was a very useful document at the beginning of my investigation. However as it became more apparent that FDD was not the right choice of process for a project of this type, it became less useful.

[6] D. Whitley, “A Genetic Algorithm Tutorial,” *Statistics and Computing,* vol. 4, no. 2, pp. 65-85, 1993.

An in depth descussion of genetic algorithms at the level understandable at a level for those new to the subject, this was a useful reference for the basic structure of a genetic algorithm as well as internal functions. It also contained some information I used refering to mutation rates and methods of mutation.

[7] A. Clarke and E. Dawson, “A Parallel Genetic Algorithm for Cryptanalysis of the Polyalphabetic Substitution Cipher,” *Cryptologia,* vol. 21, no. 2, pp. 129-138, 1997.

Not a huge amount of influence came from this paper. It again used fitness formulas based on n-gram frequencies. Used some interesting ideas revolving around running parallel parts of the genetic algorithm to attack multiple key lengths, which could have been implemented to run multiple instances of my algorithm to calculate results faster.

[8] R. Toemeh and S. Arumugam, “Applying Genetic Algorithms for Searching Key-Space of Polyalphabetic Subsititution Ciphers,” *International Arab Journal of Information Technology,* vol. 5, no. 1, pp. 87-91, 2008.

Again some useful discussion of high level genetic algorithms, and also some in depth investigation of polyalphabetic ciphers, though this was not particularly useful in this specific instance. Also used low level mathematical formulas involving n-gram frequency statistics for fitness calculation, similar to Spillman et al " Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers" which could have been useful if I was to refine fitness calculation in this genetic algorithm example.

[9] D. Whitley, “Genetic Algorithms and Evolutionary Computing,” in *Van Nostrand's Scientific Encyclopedia*, 2002.

A good reference for where the idea of genetic algorithms sit within the subject of artificial intelligence and evolutionary computing, as well as the difference between genetic algorithms and evolutionary computing. It contained useful information about the subjects of problem encoding, and crossover methods and disadvantages thereof.

[10] D. Whitley, “Genetic Algorithms and Neural Networks,” in *Genetic Algorithms in Engineering and Computier Science 3*, 1994, pp. 203-216.

A side not on my reading about differing uses of genetic algorithms, contained a little about high level genetic algorithms, but the majority of the paper was not relevant to this project.

[11] R. A. J. Matthews, “The Use of Genetic Algorithms in Cryptanalysis,” *Cryptologia,* vol. 17, no. 2, pp. 187-201, 1993.

Another old paper, this however had some quite useful information in it. It contained the basis for my fitness calculation design, as well as more information about the use of crossover and elitism in this type of genetic algorithm.

[12] B. L. Miller and D. E. Goldberg, “Genetic Algoritms, Tournament Selection and the Effects of Noise,” *Complex Systems,* vol. 9, no. 3, pp. 193-212, 1995.

A reference on tournament selection and the uses of this method of selection, advantages and disadvantages.

[13] P. K. Basavaraju, “Heuristic Search Cryptanalysis of the Zodiac 340 Cipher,” Unpublished Master's Thesis, San Jose State University, California, 2009.

A very close parallel to the attempt I was making to attempt the Zodiac ciphers. This paper took a smaller number of variables and developed them slightly further, rather than my attempt to compare a number of different combinations of operators. There was a large amount of useful investigation of the ciphers however, and the investigation using the genetic algorithm influenced my investigation process quite a lot.

[14] P. A. Diaz-Gomez and D. F. Hougen, “Initial Population for Genetic Algorithms: A Metric Approach,” in *GEM*, 2007, pp. 43-49.

An investigation into the most useful size of population for a genetic algorithm. Had useful information about the effects population size can have, as well as those factors which would have an effect on the size of population size required.

[15] M. O. Odetayo, “Optimal Population Size for Genetic Algorithms: An Investigation,” *Genetic Algorithms for Control Systems Engineering, IEE Colloquium on,* vol. 2, no. 1, 1993.

A short mathematical overview of the effect of population size on one specific type of genetic algorithms. Though the information was taken into account when running my genetic algorithm, it was unclear as to whether the data collected for this paper could be applied to all genetic algorithms, or simply the one used in the experimentation.

[16] C. Symons, Solving the Zodiac, 1st ed., C. Symons, 2009.

A concise representation of all of the evidence presented in the original investigations of the Zodiac case. Containing all of the case notes and other police documentation this was a little too detailed for my uses, but contained useful information on dates and for cross-referencing information from other sources.

[17] “Zodiac Ciphers Wiki,” 2014. [Online]. Available: http://zodiackillerciphers.com/wiki/index.php. [Accessed April 2014].

Though a community driven resource this wiki contained a large amount of useful information relating to both the case and the ciphers themselves. Much of the information could be cross-referenced with other sources and those sources cited by users were very useful, however the site itself also contained insights not found elsewhere.

[18] T. Atkins, “Substitution Cipher,” 2014. [Online]. Available: http://rumkin.com/tools/cipher/substitution.php. [Accessed 20 April 2014].

A useful tool for creating simple substitution ciphers, includes all the features I required for creating ciphers for both the background section and for use in experimentation.

[19] T. Voigt, “Zodiac Killer,” 1998. [Online]. Available: http://zodiackiller.com/. [Accessed 26 April 2014].

The most complete source of scans of the Zodiac correspondence I could find. The scans were used in both creating computer based models of the ciphers and for illustration in this report.

[20] M. Cole, “Two new theories regarding the Zodiac Case,” 10 August 2003. [Online]. Available: http://mikecole.org/zodiac/two_theories/1.2/two_theories.1.2.pdf. [Accessed 21 April 2014].

An example of where new ideas have come forward more recently about the Zodiac ciphers. Though I did not take into account any of the ideas specifically put down in this article, it was useful in the thought process of how I would attempt the ciphers myself.

[21] C. Christensen, “Simple Substitution Ciphers,” Northern Kentucky University, Cincinnati, 2006.

A useful source of the history and description of simple substitution ciphers, as well as containing much information about the currently used methods of cryptanalysis of this type of cipher. Also contained useful examples of substitution ciphers.

[22] G. J. Simmons, “Encyclopædia Britannica - Substitution Cipher,” Encyclopædia Britannica, Inc., 14 May 2013. [Online]. Available: http://www.britannica.com/EBchecked/topic/571059/substitution-cipher. [Accessed 18 April 2014].

A brief, simple definition of a substitution cipher, which was useful as a confirmed source of the information, as many other sources do not cite for this information.

[23] P. Norvig, “Natual Language Corpus Data,” 13 March 2012. [Online]. Available: http://norvig.com/ngrams/. [Accessed 27 March 2014].

A source of large scale examples of bi and tri-grams for the English language for use as a spell checking example for the English language.

[24] “N-gram tutorial,” Missouri Uniersity of Science and Techonology, [Online]. Available: http://web.mst.edu/~tauritzd/ngram/tutorial.html. [Accessed 24 March 2014].

A clear description of the definition and major uses of n-grams.

[25] F. A. Stahl, "A homophonic cipher for computational cryptography," in *National Computer Conference and exposition*, New York, 1973.

A useful paper on the creation of homophonic ciphers for use in computational cryptography. An in depth look at the history and creation of the homophonic cipher was a useful addition to my knowledge about this type of cipher. In addition it had examples of cipher creation using both the Latin alphabet and in binary. The use of binary to create homophonic ciphers could have been useful if I had modelled the Zodiac ciphers in a different fashion.

[26] "Who is the "Concerned Citizen"?," 29 December 2012. [Online]. Available: http://www.zodiackillerciphers.com/?p=224. [Accessed 29 March 2014].

A simple blog containing all the currently known information about the 'concerned citizen' the second member of the public to solve the Z408 cipher. Useful as a historical reference and source of information about the individual, but as I mostly used the Hardens solution to the Z408 information about the cipher key was mostly irrelevant.

[27] M. Obitko, "Introduction to Genetic Algorithms," 1998. [Online]. Available: http://www.obitko.com/tutorials/genetic-algorithms/index.php. [Accessed 17 January 2014].

A very useful, simple overview of the major areas of a genetic algorithm. Contained useful information about all the major functions of the algorithm, as well as suggestions about methods of mutation and crossover and other operators. Also contained some useful background knowledge about the biological background of the genetic algorithm and the origins of the ideas in computer science.

[28] F. G. Lobo and C. F. Lima, "A review of adaptive population sizing schemems in genetic algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference*, Washington, D.C., 2005.

As an investigation into the use of a population size that changed, this was a very useful article. However as I decided to design a genetic algorithm with a fixed size population, its usefulness was reduced somewhat.

[29] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," University of Illinois, Urbana, 1991.

A more in depth look at the use of tournament selection as well as its advantages and disadvantages. The paper also looks at a number of other selection operator used in genetic algorithms which I did not discuss in this document.

[30] "Letter frequency (English)," Algorithmy, 2008. [Online]. Available: http://en.algoritmy.net/article/40379/Letter-frequency-English. [Accessed 27 April 2014].

A source of the letter frequencies of the general written and spoken English language.

[31] "Stochastic universal sampling," Wikimedia Foundation Inc., 24 May 2013. [Online]. Available: http://en.wikipedia.org/wiki/Stochastic_universal_sampling. [Accessed 28 April 2014].

Source of a useful diagram explaining stochastic universal sampling which could not be sourced from anywhere else. Seemed to be the best documented diagram of the information

[32] “Fitness proportionate selection,” Wikimedia Foundation Inc., 5 February 2014. [Online]. Available: http://en.wikipedia.org/wiki/Fitness_proportionate_selection. [Accessed 28 April 2014].

Source of a useful diagram explaining fitness proportionate selection which could not be sourced from anywhere else. Seemed to be the best documented diagram of the information.

[33] “Mono Cross platform, open source .NET development framework,” Zamarin, [Online]. Available: http://www.mono-project.com/Main_Page. [Accessed 10 February 2014].

Cross platform c# framework which could have been used if necessary to move the development onto another operating system.

[34] “GitHub,” GitHub Inc., 2014. [Online]. Available: https://github.com/. [Accessed 20 January 2014].

One of the two options, and the eventual choice for project version control.

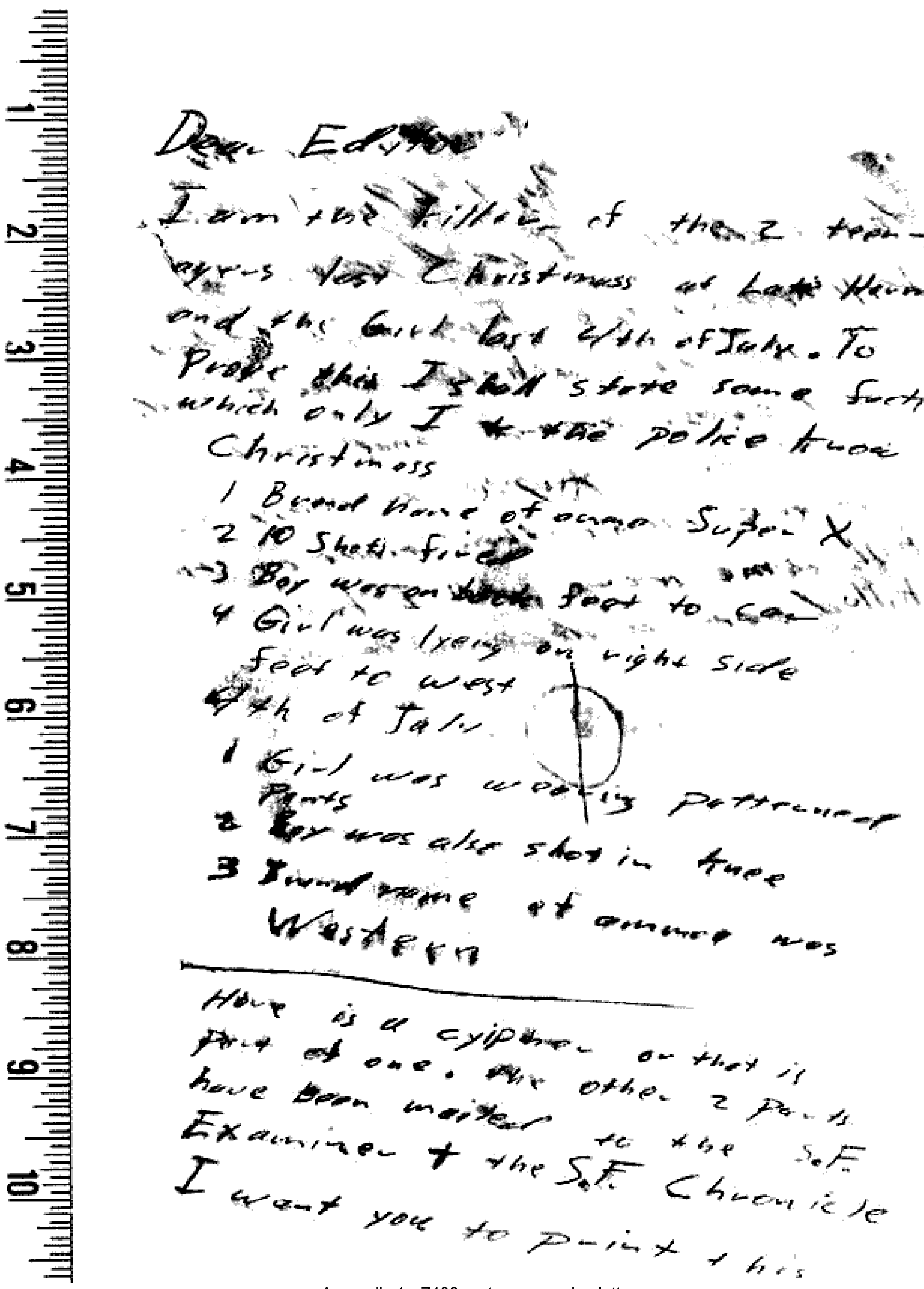
[35] “Apache Subversion,” Apache, 2014. [Online]. Available: http://subversion.apache.org/. [Accessed 20 January 2014].

One of two options for project version control.

[36] P. E. Black, “Fisher-Yates shuffle,” 23 May 2011. [Online]. Available: http://www.nist.gov/dads/HTML/fisherYatesShuffle.html. [Accessed 29 April 2014].

A description of the sudo random shuffle I used in order to create large numbers of unique chromosomes for the substitution cipher.  A very simple description with links to further information and implentations.

## [Appendix A] Zodiac Covering Letters



Dear Editor

I am the killer of the 2 teen-agers last Christmass at Lake Herman and the Girl last 4th of July. To prove this I shall state some facts which only I + the police know

Christmass

1 Brand name of ammo Super X

2 10 shots fired

3 Boy was on back feet to car

4 Girl was lyeing on right side feet to west

4th of July

1 Girl was wearing patterned pants

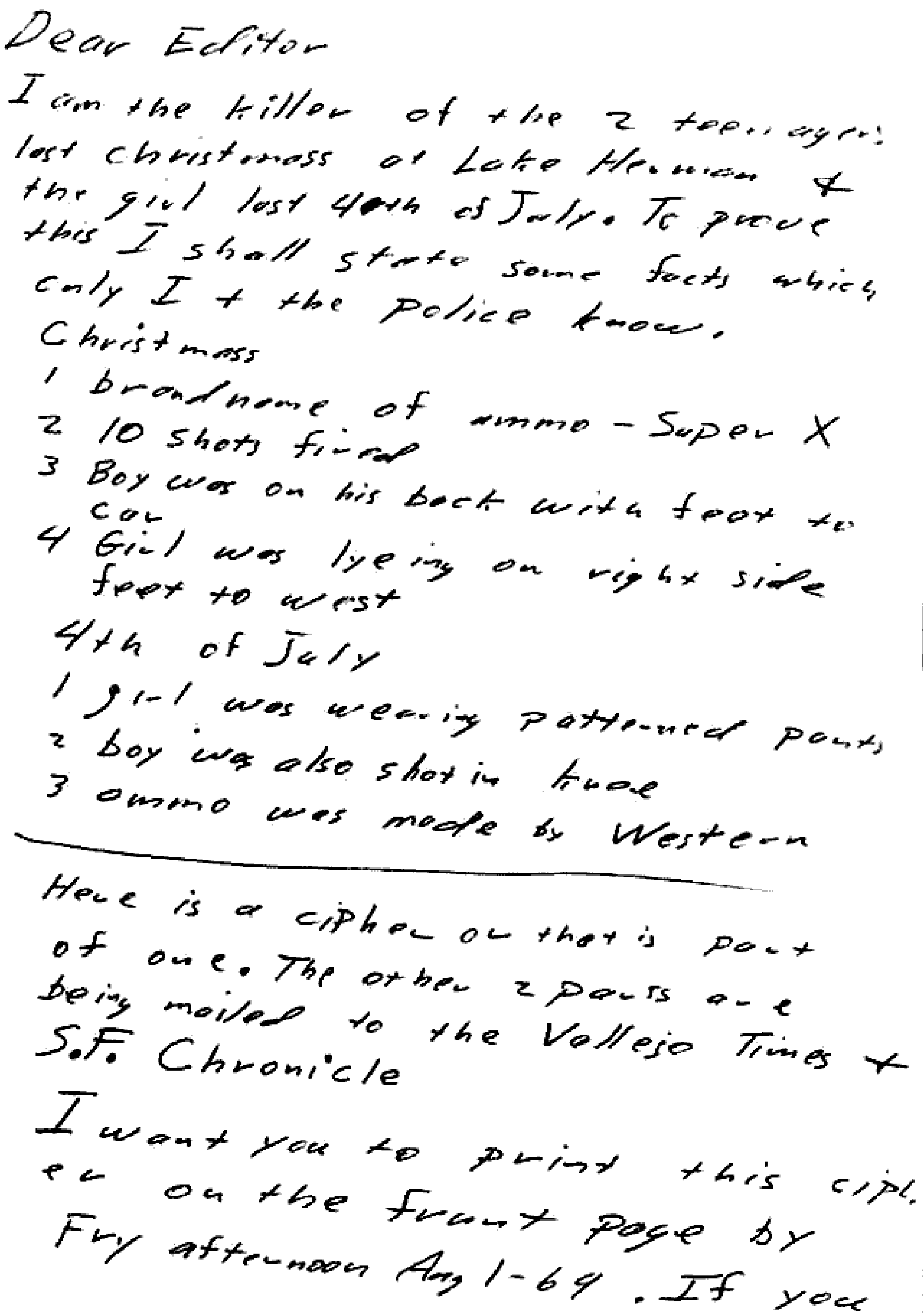
2 Boy was also shot in knee

3 Brand name of ammo was Western

Here is a cyipher or that is part of one. The other 2 parts have been mailed to the S.F. Examiner + the S.F. Chronicle

I want you to print this

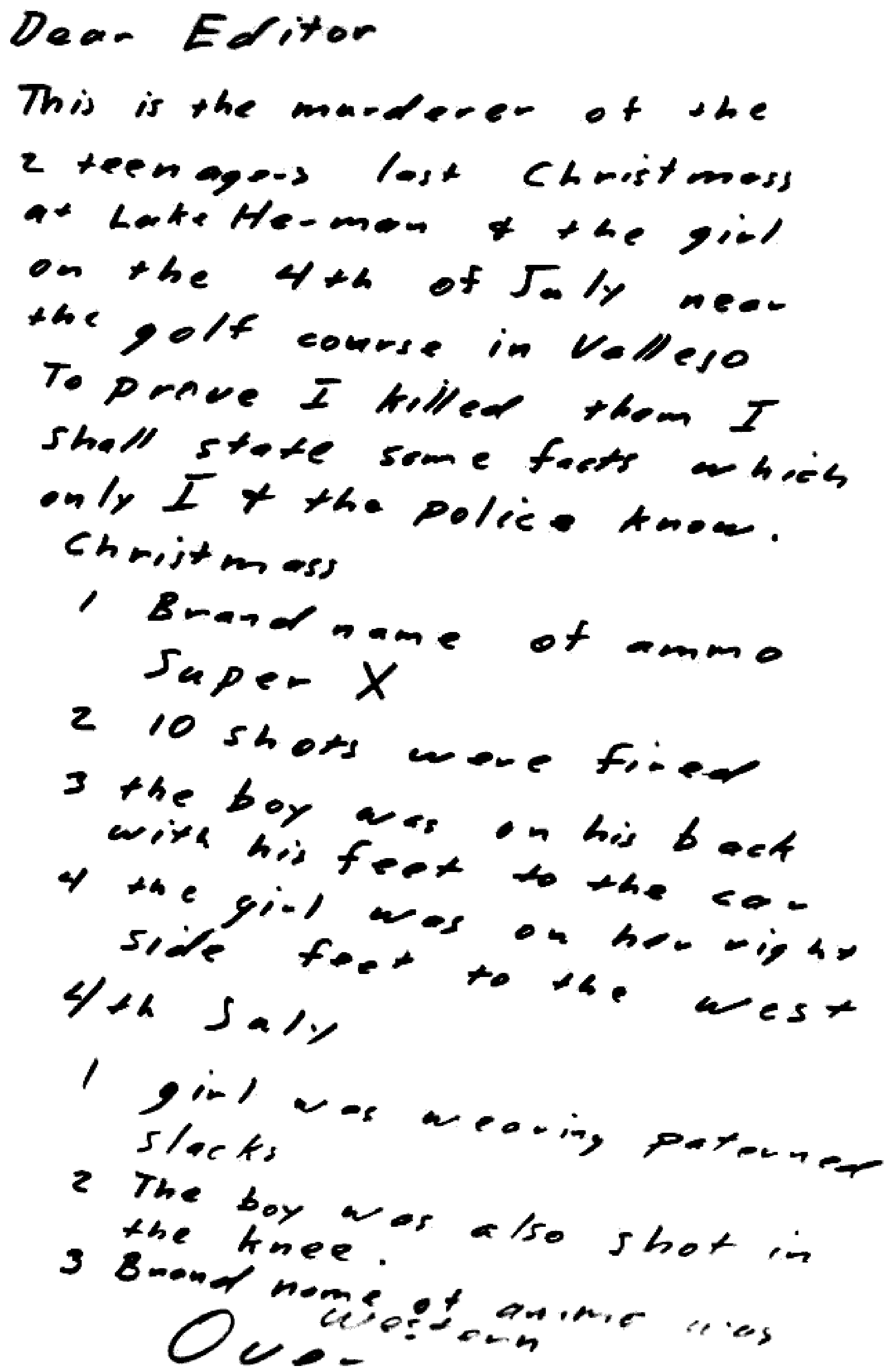
Appendix 1 - Z408 part one covering letter



Dear Editor
I am the killer of the 2 teenagers
last Christmass at Lake Herman &
the girl last 4th of July. To prove
this I shall state some facts which
only I & the police know.
Christmass
1 brandname of ammo - Super X
2 10 shots fired
3 Boy was on his back with feet to
car
4 Girl was lyeing on right side
feet to west
4th of July
1 girl was wearing patterned pants
2 boy was also shot in knee
3 ammo was made by Western

Here is a cipher or that is part
of one. The other 2 parts are
being mailed to the Vallejo Times &
S.F. Chronicle
I want you to print this ciph
er on the front page by
Fry afternoon Aug 1-69. If you

Appendix 2 - Z408 part two covering letter



Dear Editor

This is the murderer of the
2 teenagers last Christmass
at Lake Herman & the girl
on the 4th of July near
the golf course in Vallejo
To prove I killed them I
Shall state some facts which
only I & the police know.

Christmass

1 Brand name of ammo
Super X

2 10 shots were fired

3 the boy was on his back
with his feet to the car

4 the girl was on her right
side feet to the west

4th July

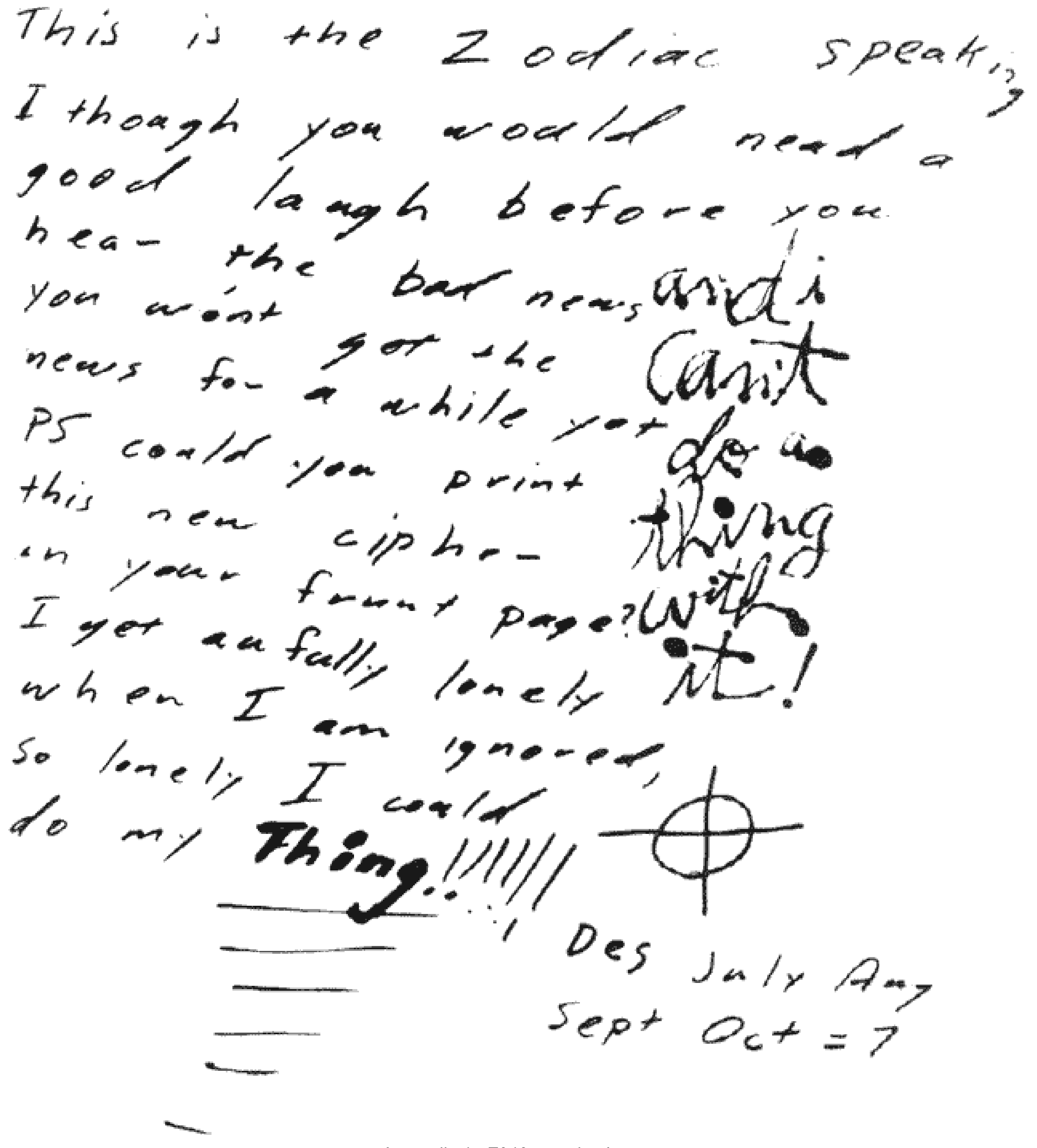
1 girl was wearing paterned
slacks

2 The boy was also shot in
the knee.

3 Brand name of ammo was
Western

Over

Appendix 3 - Z408 part three covering letter

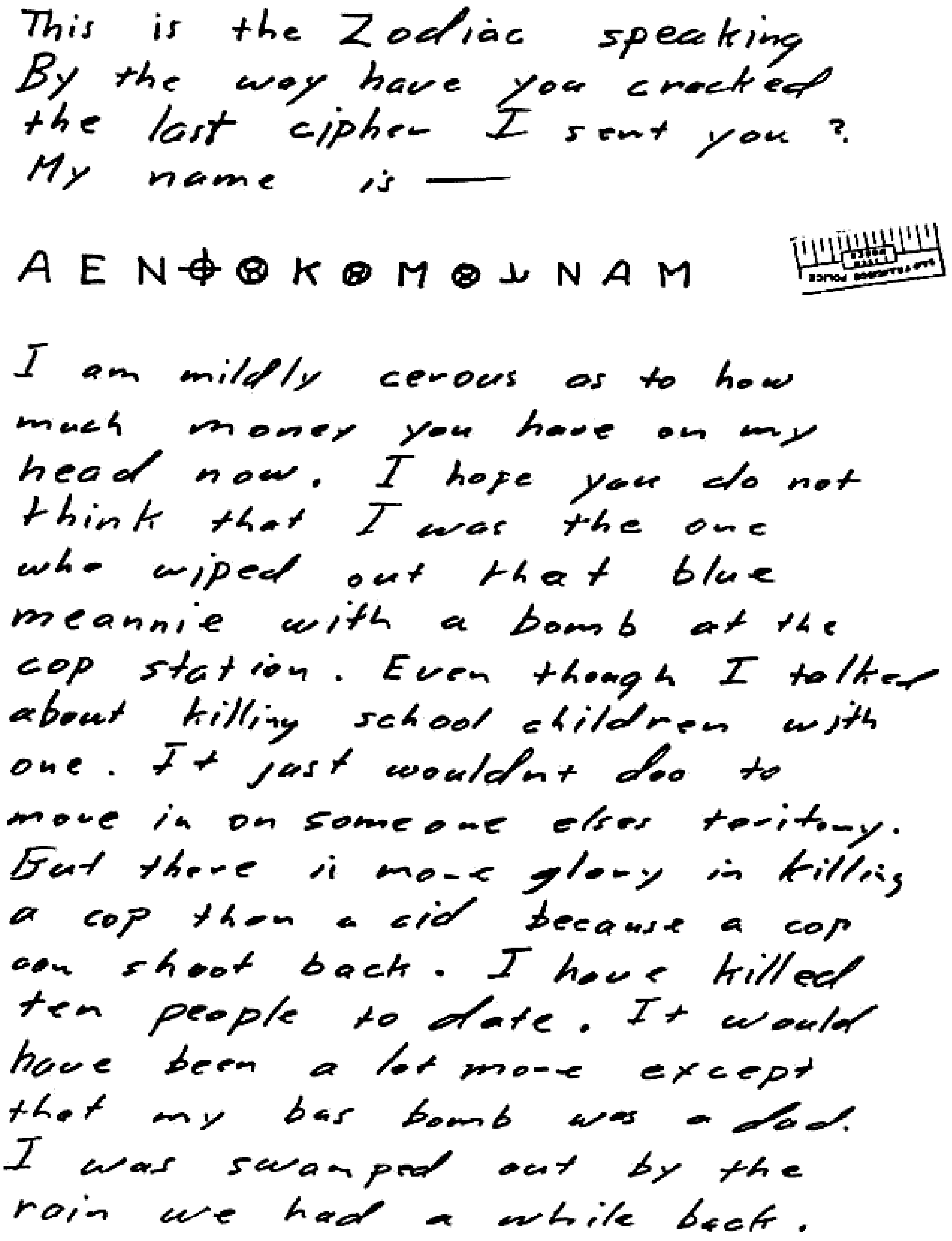


This is the Zodiac speaking
I though you would need a
good laugh before you
hear the
bad news
you wont get the
news for a while yet
PS could you print
this new cipher
in your frunt page?
I get awfully lonely
when I am ignored,
So lonely I could
do my Thing!!!!!!!

and i cant do a thing with it!

Des July Aug
Sept Oct = 7

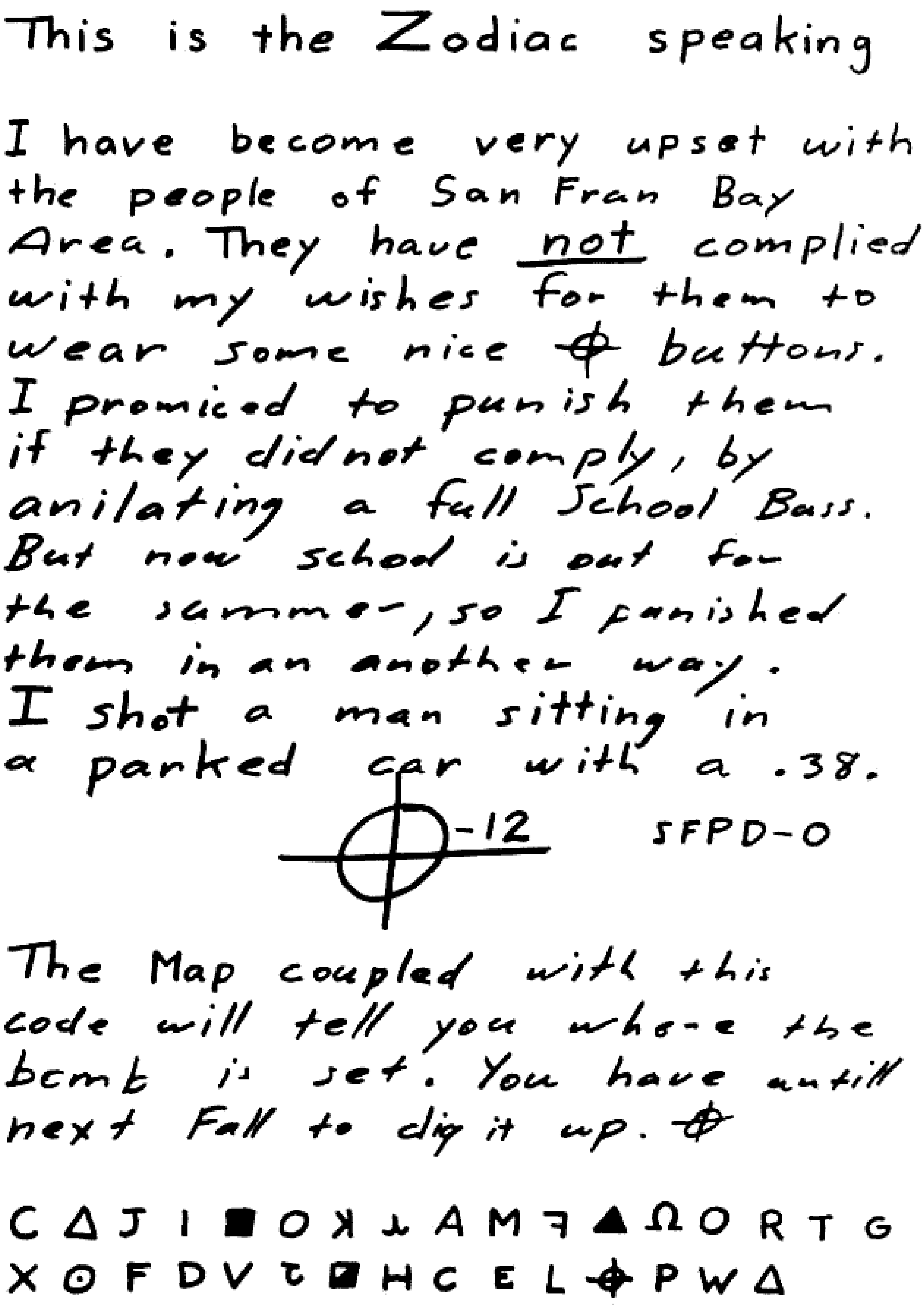
Appendix 4 - Z340 covering letter


This is the Zodiac speaking
By the way have you cracked
the last cipher I sent you ?.
My name is —

A E N ⊕ ⊗ K ⊗ M ⊗ ⊥ N A M

I am mildly cerous as to how
much money you have on my
head now. I hope you do not
think that I was the one
who wiped out that blue
meannie with a bomb at the
cop station. Even though I talked
about killing school children with
one. It just wouldnt doo to
move in on someone elses teritory.
But there is more glory in killing
a cop than a cid because a cop
can shoot back. I have killed
ten people to date. It would
have been a lot more except
that my bus bomb was a dud.
I was swamped out by the
rain we had a while back.

Appendix 5 - Z13 and covering letter



This is the Zodiac speaking

I have become very upset with the people of San Fran Bay Area. They have <u>not</u> complied with my wishes for them to wear some nice ⊕ buttons. I promiced to punish them if they didnot comply, by anilating a full School Buss. But now school is out for the summer, so I punished them in an another way. I shot a man sitting in a parked car with a .38.

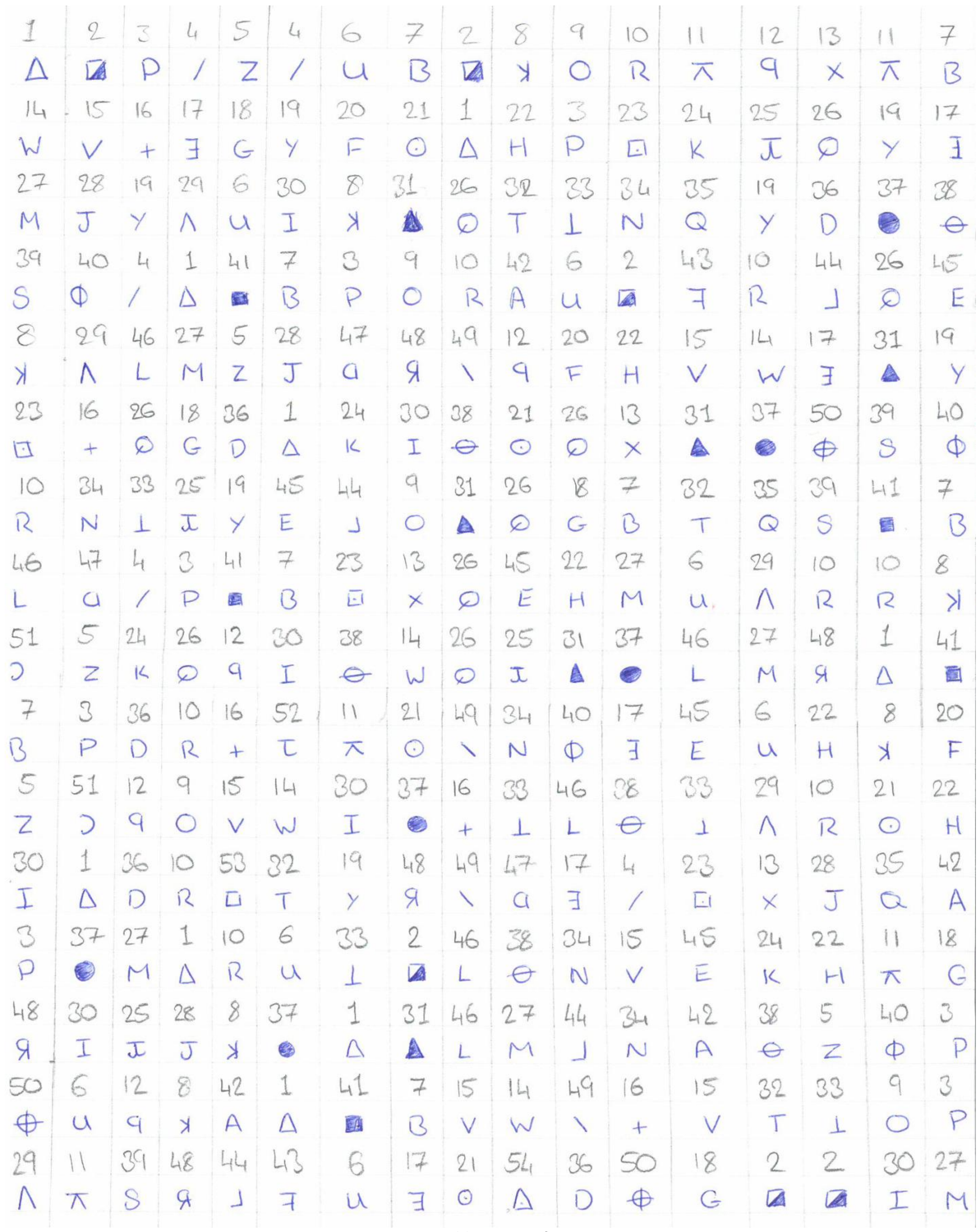
⊕-12 SFPD-0

The Map coupled with this code will tell you where the bomb is set. You have untill next Fall to dig it up. ⊕

C Δ J I ■ O ꓘ ⅄ A M ⅂ ▲ Ω O R T G
X ⊙ F D V ꞇ ◩ H C E L ⊕ P W Δ

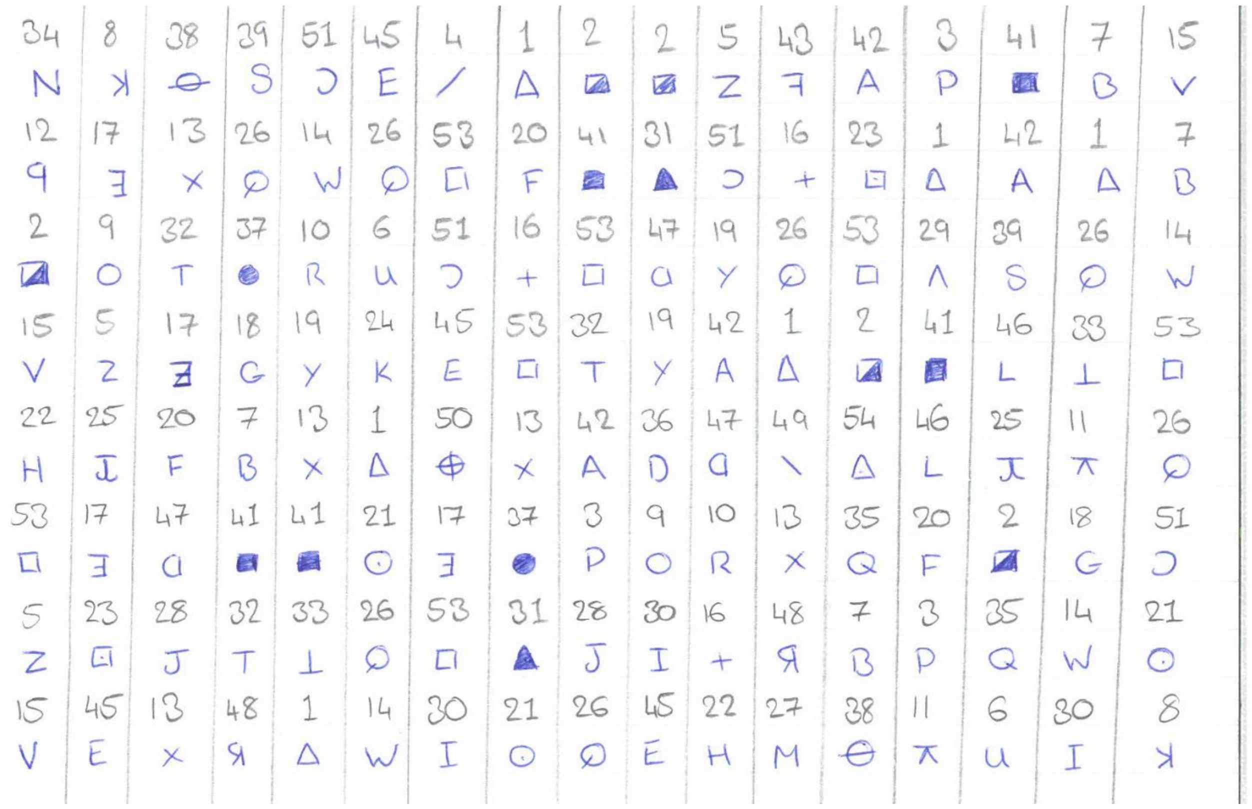
Appendix 6 - Z32 and covering letter

## [Appendix B] Zodiac cipher numeric form working



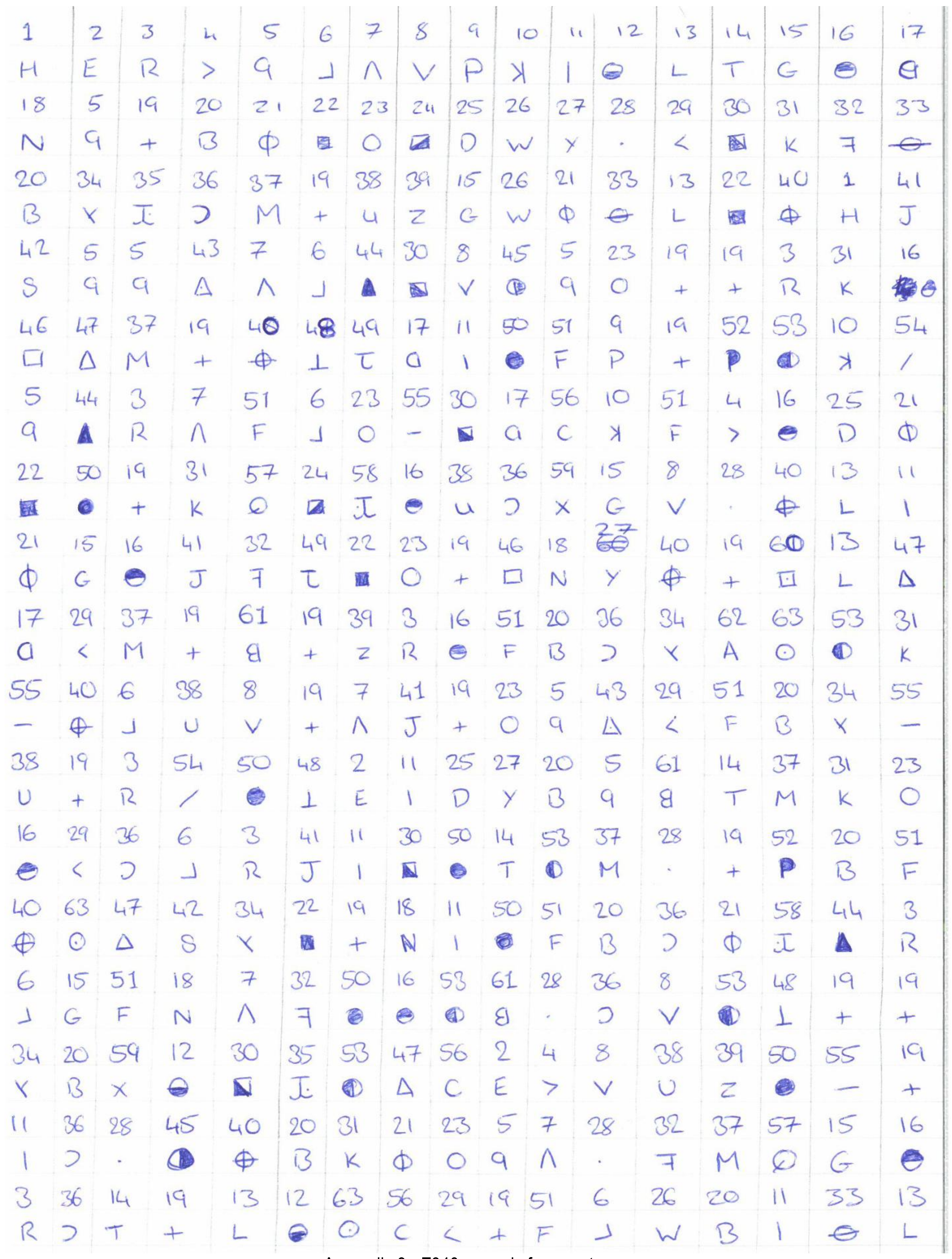
| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 4 | 6 | 7 | 2 | 8 | 9 | 10 | 11 | 12 | 13 | 11 | 7 |
| Δ | ◪ | P | / | Z | / | U | B | ◪ | ꓘ | O | R | ⊼ | 9 | X | ⊼ | B |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 1 | 22 | 3 | 23 | 24 | 25 | 26 | 19 | 17 |
| W | V | + | Ǝ | G | Y | F | ⊙ | Δ | H | P | ⊡ | K | Ɉ | ⌀ | Y | Ǝ |
| 27 | 28 | 19 | 29 | 6 | 30 | 8 | 31 | 26 | 32 | 33 | 34 | 35 | 19 | 36 | 37 | 38 |
| M | J | Y | Λ | U | I | ꓘ | ▲ | ⌀ | T | ⊥ | N | Q | Y | D | ● | ⊖ |
| 39 | 40 | 4 | 1 | 41 | 7 | 3 | 9 | 10 | 42 | 6 | 2 | 43 | 10 | 44 | 26 | 45 |
| S | Φ | / | Δ | ■ | B | P | O | R | A | U | ◪ | ꟻ | R | ⅃ | ⌀ | E |
| 8 | 29 | 46 | 27 | 5 | 28 | 47 | 48 | 49 | 12 | 20 | 22 | 15 | 14 | 17 | 31 | 19 |
| ꓘ | Λ | L | M | Z | J | ɑ | Я | \ | 9 | F | H | V | W | Ǝ | ▲ | Y |
| 23 | 16 | 26 | 18 | 36 | 1 | 24 | 30 | 38 | 21 | 26 | 13 | 31 | 37 | 50 | 39 | 40 |
| ⊡ | + | ⌀ | G | D | Δ | K | I | ⊖ | ⊙ | ⌀ | X | ▲ | ● | ⊕ | S | Φ |
| 10 | 34 | 33 | 25 | 19 | 45 | 44 | 9 | 31 | 26 | 18 | 7 | 32 | 35 | 39 | 41 | 7 |
| R | N | ⊥ | Ɉ | Y | E | ⅃ | O | ▲ | ⌀ | G | B | T | Q | S | ■ | B |
| 46 | 47 | 4 | 3 | 41 | 7 | 23 | 13 | 26 | 45 | 22 | 27 | 6 | 29 | 10 | 10 | 8 |
| L | ɑ | / | P | ■ | B | ⊡ | X | ⌀ | E | H | M | U | Λ | R | R | ꓘ |
| 51 | 5 | 24 | 26 | 12 | 30 | 38 | 14 | 26 | 25 | 31 | 37 | 46 | 27 | 48 | 1 | 41 |
| Ɔ | Z | K | ⌀ | 9 | I | ⊖ | W | ⌀ | Ɉ | ▲ | ● | L | M | Я | Δ | ■ |
| 7 | 3 | 36 | 10 | 16 | 52 | 11 | 21 | 49 | 34 | 40 | 17 | 45 | 6 | 22 | 8 | 20 |
| B | P | D | R | + | Ꚍ | ⊼ | ⊙ | \ | N | Φ | Ǝ | E | U | H | ꓘ | F |
| 5 | 51 | 12 | 9 | 15 | 14 | 30 | 37 | 16 | 33 | 46 | 38 | 33 | 29 | 10 | 21 | 22 |
| Z | Ɔ | 9 | O | V | W | I | ● | + | ⊥ | L | ⊖ | ⊥ | Λ | R | ⊙ | H |
| 30 | 1 | 36 | 10 | 53 | 32 | 19 | 48 | 49 | 47 | 17 | 4 | 23 | 13 | 28 | 35 | 42 |
| I | Δ | D | R | ◻ | T | Y | Я | \ | ɑ | Ǝ | / | ⊡ | X | J | Q | A |
| 3 | 37 | 27 | 1 | 10 | 6 | 33 | 2 | 46 | 38 | 34 | 15 | 45 | 24 | 22 | 11 | 18 |
| P | ● | M | Δ | R | U | ⊥ | ◪ | L | ⊖ | N | V | E | K | H | ⊼ | G |
| 48 | 30 | 25 | 28 | 8 | 37 | 1 | 31 | 46 | 27 | 44 | 34 | 42 | 38 | 5 | 40 | 3 |
| Я | I | Ɉ | J | ꓘ | ● | Δ | ▲ | L | M | ⅃ | N | A | ⊖ | Z | Φ | P |
| 50 | 6 | 12 | 8 | 42 | 1 | 41 | 7 | 15 | 14 | 49 | 16 | 15 | 32 | 33 | 9 | 3 |
| ⊕ | U | 9 | ꓘ | A | Δ | ■ | B | V | W | \ | + | V | T | ⊥ | O | P |
| 29 | 11 | 39 | 48 | 44 | 43 | 6 | 17 | 21 | 54 | 36 | 50 | 18 | 2 | 2 | 30 | 27 |
| Λ | ⊼ | S | Я | ⅃ | ꟻ | U | Ǝ | ⊙ | Δ | D | ⊕ | G | ◪ | ◪ | I | M |

Appendix 7 - Z408 numeric form part one



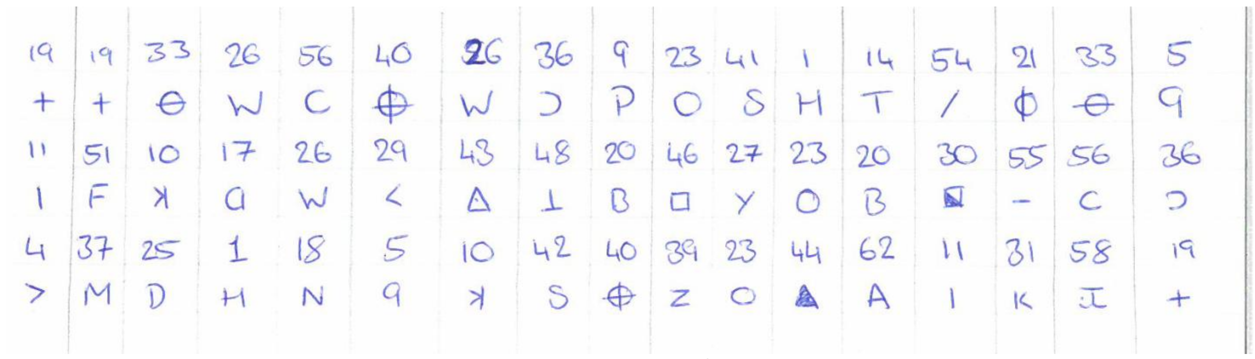
| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 34 | 8 | 38 | 39 | 51 | 45 | 4 | 1 | 2 | 2 | 5 | 43 | 42 | 3 | 41 | 7 | 15 |
| N | ꓘ | ⊖ | S | Ↄ | E | / | Δ | ◪ | ◪ | Z | ꟻ | A | P | ■ | B | V |
| 12 | 17 | 13 | 26 | 14 | 26 | 53 | 20 | 41 | 31 | 51 | 16 | 23 | 1 | 42 | 1 | 7 |
| 9 | Ǝ | X | ⌀ | W | ⌀ | ◫ | F | ■ | ▲ | Ↄ | + | ⊡ | Δ | A | Δ | B |
| 2 | 9 | 32 | 37 | 10 | 6 | 51 | 16 | 53 | 47 | 19 | 26 | 53 | 29 | 39 | 26 | 14 |
| ◪ | O | T | ● | R | U | Ↄ | + | ◫ | a | Y | ⌀ | ◫ | Λ | S | ⌀ | W |
| 15 | 5 | 17 | 18 | 19 | 24 | 45 | 53 | 32 | 19 | 42 | 1 | 2 | 41 | 46 | 33 | 53 |
| V | Z | Ƨ | G | Y | K | E | ◫ | T | Y | A | Δ | ◪ | ■ | L | ⊥ | ◫ |
| 22 | 25 | 20 | 7 | 13 | 1 | 50 | 13 | 42 | 36 | 47 | 49 | 54 | 46 | 25 | 11 | 26 |
| H | J | F | B | X | Δ | ⊕ | X | A | D | a | \ | Δ | L | J | ⊼ | ⌀ |
| 53 | 17 | 47 | 41 | 41 | 21 | 17 | 37 | 3 | 9 | 10 | 13 | 35 | 20 | 2 | 18 | 51 |
| ◫ | Ǝ | a | ■ | ■ | ⊙ | Ǝ | ● | P | O | R | X | Q | F | ◪ | G | Ↄ |
| 5 | 23 | 28 | 32 | 33 | 26 | 53 | 31 | 28 | 30 | 16 | 48 | 7 | 3 | 35 | 14 | 21 |
| Z | ⊡ | J | T | ⊥ | ⌀ | ◫ | ▲ | J | I | + | Я | B | P | Q | W | ⊙ |
| 15 | 45 | 13 | 48 | 1 | 14 | 30 | 21 | 26 | 45 | 22 | 27 | 38 | 11 | 6 | 30 | 8 |
| V | E | X | Я | Δ | W | I | ⊙ | ⌀ | E | H | M | ⊖ | ⊼ | U | I | ꓘ |

Appendix 8 - Z408 numeric form part two



| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| H | E | R | > | q | ⅃ | Λ | V | P | ꓘ | I | ⊖ | L | T | G | ● | ɑ |
| 18 | 5 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 |
| N | 9 | + | B | ɸ | ▣ | O | ◪ | D | W | Y | · | < | ▨ | K | ꟻ | ⊖ |
| 20 | 34 | 35 | 36 | 37 | 19 | 38 | 39 | 15 | 26 | 21 | 33 | 13 | 22 | 40 | 1 | 41 |
| B | Y | I | Ↄ | M | + | U | Z | G | W | ɸ | ⊖ | L | ▣ | ⊕ | H | J |
| 42 | 5 | 5 | 43 | 7 | 6 | 44 | 30 | 8 | 45 | 5 | 23 | 19 | 19 | 3 | 31 | 16 |
| S | 9 | 9 | Δ | Λ | ⅃ | ▲ | ▨ | V | ⊕ | 9 | O | + | + | R | K | ● |
| 46 | 47 | 37 | 19 | 40 | 48 | 49 | 17 | 11 | 50 | 51 | 9 | 19 | 52 | 53 | 10 | 54 |
| ⊐ | Δ | M | + | ⊕ | ⊥ | Ƭ | ɑ | I | ● | F | P | + | ▶ | ◐ | ꓘ | / |
| 5 | 44 | 3 | 7 | 51 | 6 | 23 | 55 | 30 | 17 | 56 | 10 | 51 | 4 | 16 | 25 | 21 |
| 9 | ▲ | R | Λ | F | ⅃ | O | – | ▨ | ɑ | C | ꓘ | F | > | ● | D | ɸ |
| 22 | 50 | 19 | 31 | 57 | 24 | 58 | 16 | 38 | 36 | 59 | 15 | 8 | 28 | 40 | 13 | 11 |
| ▣ | ● | + | K | Ø | ◪ | I | ● | U | Ↄ | X | G | V | · | ⊕ | L | I |
| 21 | 15 | 16 | 41 | 32 | 49 | 22 | 23 | 19 | 46 | 18 | 27 60 | 40 | 19 | 60 | 13 | 47 |
| ɸ | G | ● | J | ꟻ | Ƭ | ▣ | O | + | ⊐ | N | Y | ⊕ | + | ⊡ | L | Δ |
| 17 | 29 | 37 | 19 | 61 | 19 | 39 | 3 | 16 | 51 | 20 | 36 | 34 | 62 | 63 | 53 | 31 |
| ɑ | < | M | + | 8 | + | Z | R | ● | F | B | Ↄ | Y | A | ⊙ | ◐ | K |
| 55 | 40 | 6 | 38 | 8 | 19 | 7 | 41 | 19 | 23 | 5 | 43 | 29 | 51 | 20 | 34 | 55 |
| – | ⊕ | ⅃ | U | V | + | Λ | J | + | O | 9 | Δ | < | F | B | Y | – |
| 38 | 19 | 3 | 54 | 50 | 48 | 2 | 11 | 25 | 27 | 20 | 5 | 61 | 14 | 37 | 31 | 23 |
| U | + | R | / | ● | ⊥ | E | I | D | Y | B | 9 | 8 | T | M | K | O |
| 16 | 29 | 36 | 6 | 3 | 41 | 11 | 30 | 50 | 14 | 53 | 37 | 28 | 19 | 52 | 20 | 51 |
| ● | < | Ↄ | ⅃ | R | J | I | ▨ | ● | T | ◐ | M | · | + | ▶ | B | F |
| 40 | 63 | 47 | 42 | 34 | 22 | 19 | 18 | 11 | 50 | 51 | 20 | 36 | 21 | 58 | 44 | 3 |
| ⊕ | ⊙ | Δ | S | Y | ▣ | + | N | I | ● | F | B | Ↄ | ɸ | I | ▲ | R |
| 6 | 15 | 51 | 18 | 7 | 32 | 50 | 16 | 53 | 61 | 28 | 36 | 8 | 53 | 48 | 19 | 19 |
| ⅃ | G | F | N | Λ | ꟻ | ● | ● | ◐ | 8 | · | Ↄ | V | ◐ | ⊥ | + | + |
| 34 | 20 | 59 | 12 | 30 | 35 | 53 | 47 | 56 | 2 | 4 | 8 | 38 | 39 | 50 | 55 | 19 |
| Y | B | X | ⊖ | ▨ | I | ◐ | Δ | C | E | > | V | U | Z | ● | – | + |
| 11 | 36 | 28 | 45 | 40 | 20 | 31 | 21 | 23 | 5 | 7 | 28 | 32 | 37 | 57 | 15 | 16 |
| I | Ↄ | · | ◐ | ⊕ | B | K | ɸ | O | 9 | Λ | · | ꟻ | M | Ø | G | ● |
| 3 | 36 | 14 | 19 | 13 | 12 | 63 | 56 | 29 | 19 | 51 | 6 | 26 | 20 | 11 | 33 | 13 |
| R | Ↄ | T | + | L | ⊖ | ⊙ | C | < | + | F | ⅃ | W | B | I | ⊖ | L |

Appendix 9 - Z340 numeric form part one



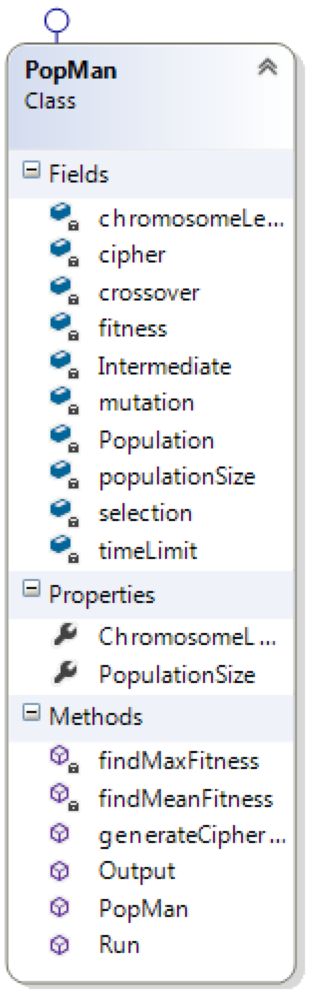
| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 19 | 19 | 33 | 26 | 56 | 40 | 26 | 36 | 9 | 23 | 41 | 1 | 14 | 54 | 21 | 33 | 5 |
| + | + | ⊖ | W | C | ⊕ | W | ⊃ | P | O | S | H | T | / | ϕ | ⊖ | 9 |
| 11 | 51 | 10 | 17 | 26 | 29 | 43 | 48 | 20 | 46 | 27 | 23 | 20 | 30 | 55 | 56 | 36 |
| I | F | ꓘ | a | W | < | Δ | ⊥ | B | □ | Y | O | B | ◪ | - | C | ⊃ |
| 4 | 37 | 25 | 1 | 18 | 5 | 10 | 42 | 40 | 39 | 23 | 44 | 62 | 11 | 31 | 58 | 19 |
| > | M | D | H | N | 9 | ꓘ | S | ⊕ | Z | O | ▲ | A | I | K | ⌶ | + |

Appendix 10 - Z340 numeric form part two

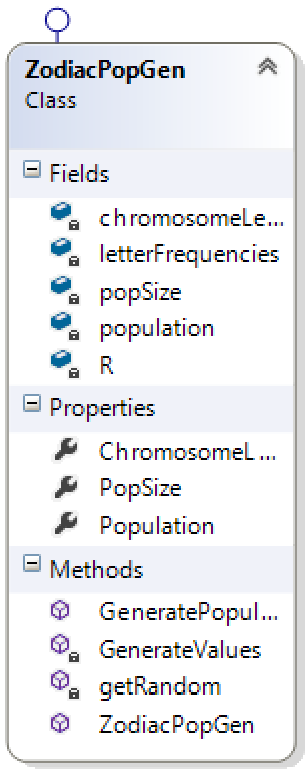
## [Appendix C] Example of N-gram Values

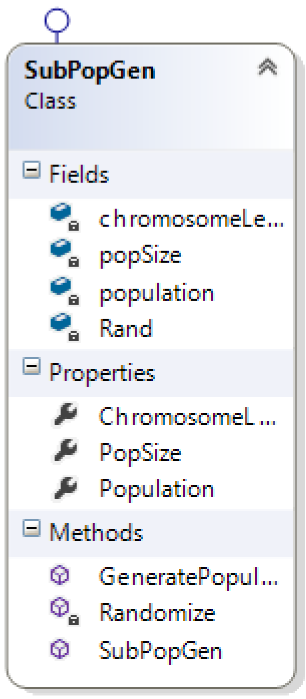
| | |
|---|---|
| in | 134812613554 |
| th | 133210262170 |
| er | 119214789533 |
| re | 108669181717 |
| he | 106498528786 |
| an | 105422467512 |
| on | 100461773921 |
| es | 97326307798 |
| or | 85998433585 |
| te | 80948650956 |
| at | 80609883139 |
| ti | 79834588969 |
| st | 78728300226 |
| en | 77314896004 |
| nt | 71720202984 |
| ar | 68756067210 |
| to | 68548751664 |
| nd | 68018862077 |
| al | 64032793037 |
| it | 60786494374 |
| se | 58165976437 |
| ed | 57514469508 |
| is | 56531345460 |
| ea | 55939058323 |
| ng | 54671896948 |
| ou | 52913816186 |
| le | 52533619401 |
| co | 52490369344 |
| me | 50958591639 |
| ne | 48426255131 |
| ri | 48154583840 |
| ro | 46608666502 |
| de | 46344292825 |
| ra | 46135420707 |
| io | 45866820943 |
| ic | 45663138305 |
| li | 45239121467 |
| of | 44948393239 |
| as | 43550945165 |
| et | 42812689527 |
| ve | 42386634729 |
| ta | 42344542093 |
| si | 41483080358 |
| ha | 41206270659 |
| ma | 40261070734 |
| ec | 39836916955 |
| om | 38270823174 |
| ce | 37978263270 |
| el | 37931534499 |
| ll | 36610231690 |
| ca | 35964886206 |
| ur | 35957660877 |
| la | 35279652135 |
| ch | 34170408619 |
| hi | 34041908708 |
| di | 33302437545 |

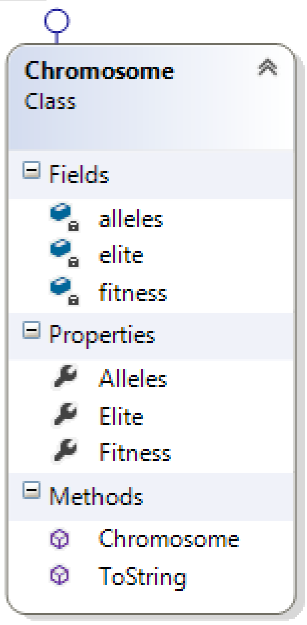
## [Appendix D] UML Diagrams



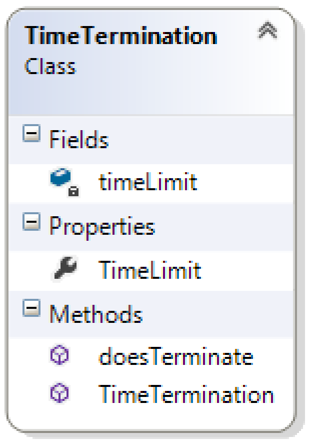
Appendix 11 - Population management



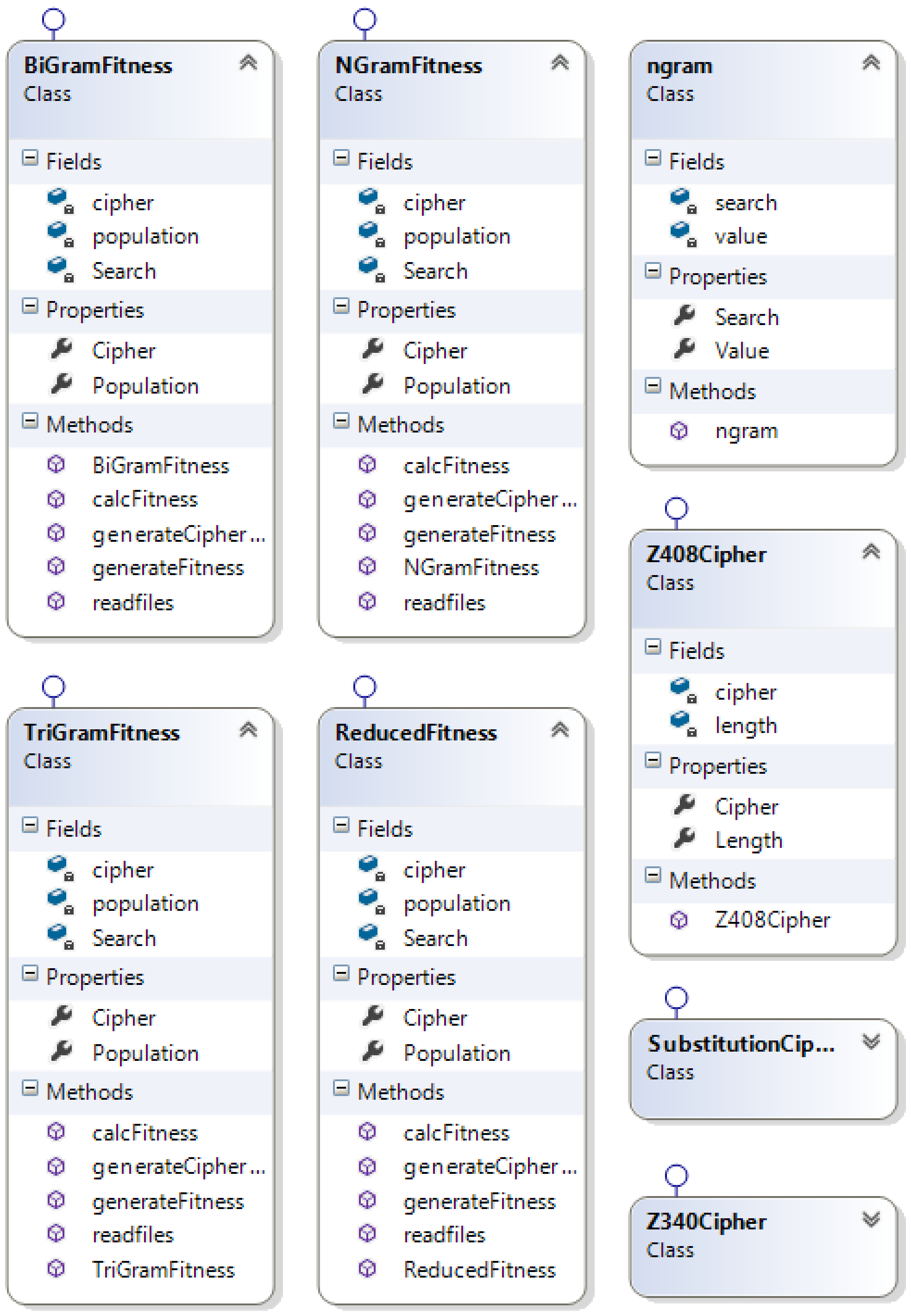




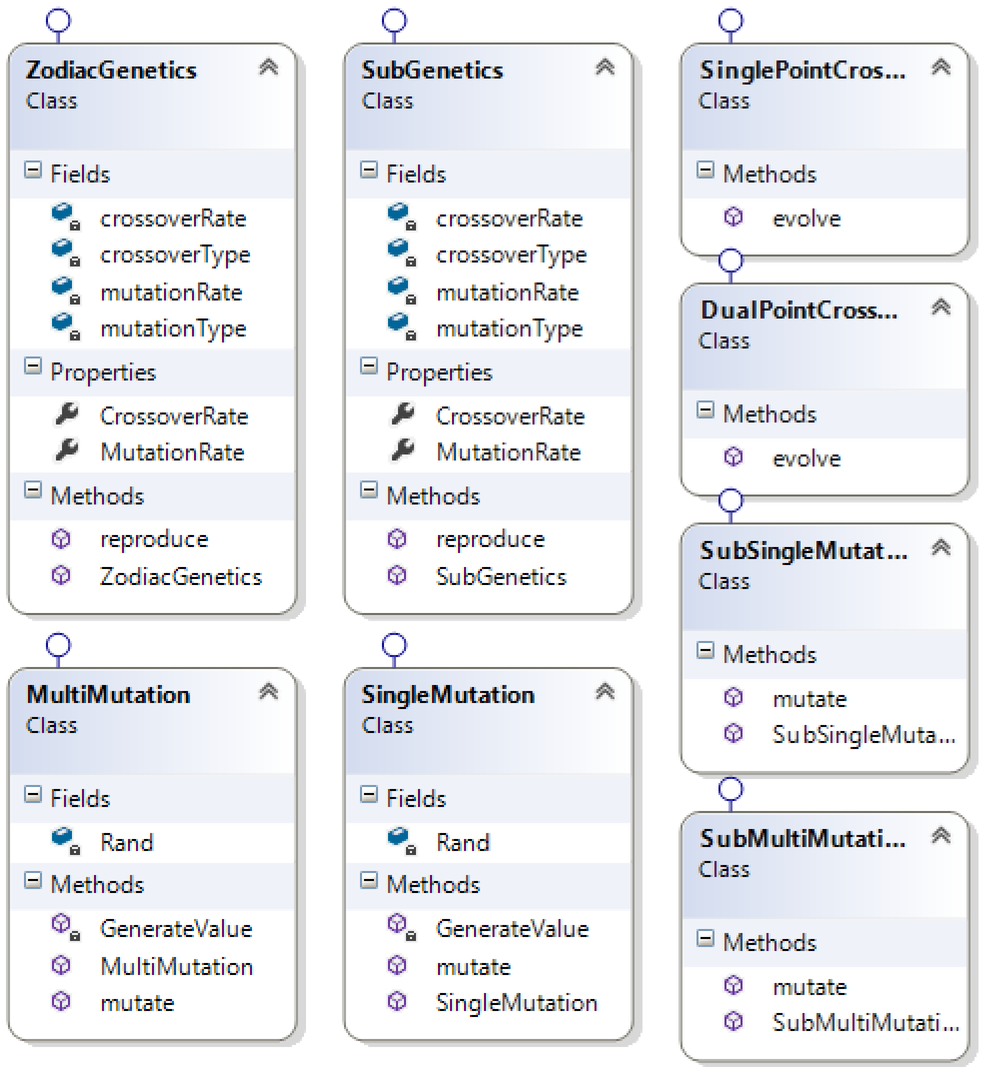
Appendix 12 - Population Generation



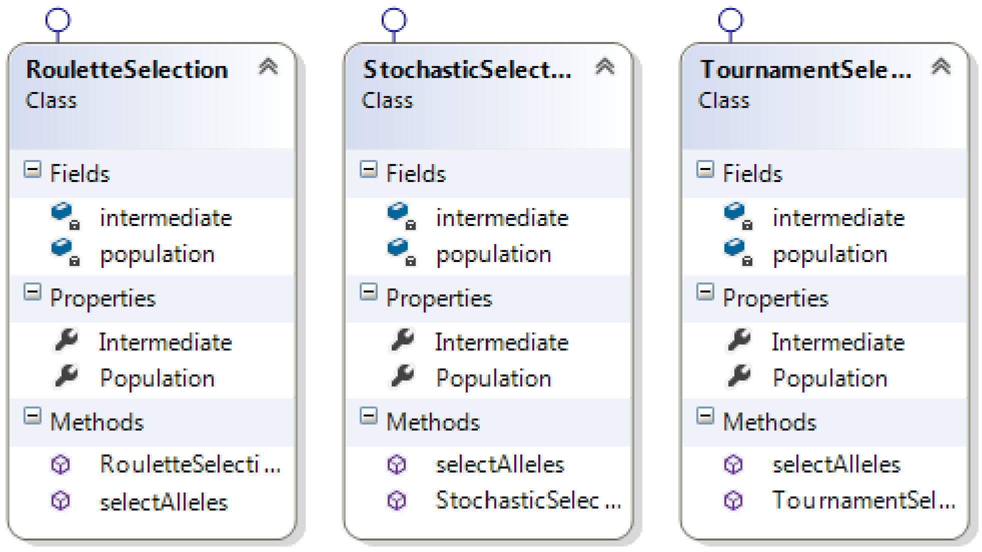
Appendix 13 - Termination



Appendix 14 - Fitness

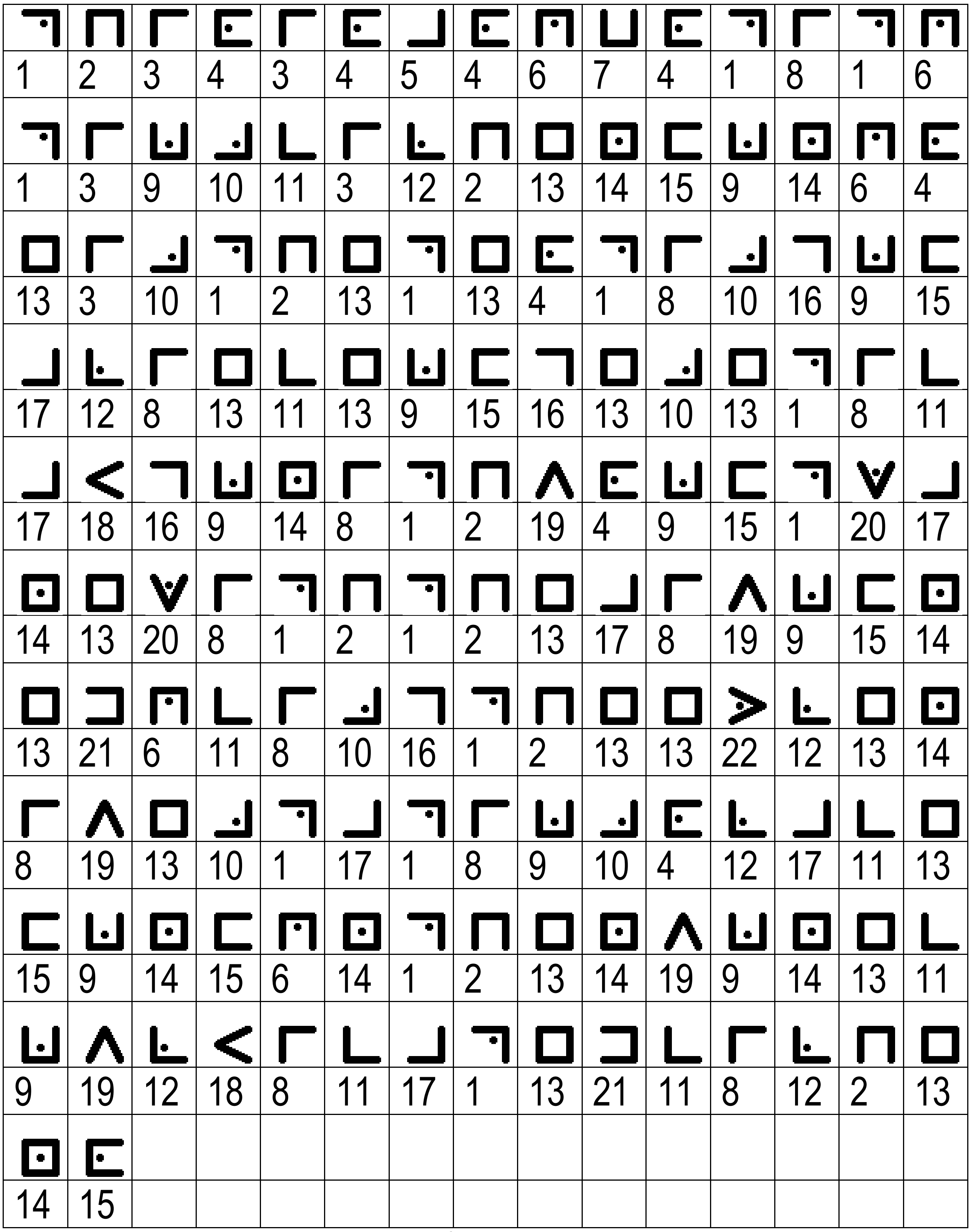


Appendix 16 - Genetics



Appendix 15 - Selection

## [Appendix E] Substitution Cipher numerical model



| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 3 | 4 | 5 | 4 | 6 | 7 | 4 | 1 | 8 | 1 | 6 |
| 1 | 3 | 9 | 10 | 11 | 3 | 12 | 2 | 13 | 14 | 15 | 9 | 14 | 6 | 4 |
| 13 | 3 | 10 | 1 | 2 | 13 | 1 | 13 | 4 | 1 | 8 | 10 | 16 | 9 | 15 |
| 17 | 12 | 8 | 13 | 11 | 13 | 9 | 15 | 16 | 13 | 10 | 13 | 1 | 8 | 11 |
| 17 | 18 | 16 | 9 | 14 | 8 | 1 | 2 | 19 | 4 | 9 | 15 | 1 | 20 | 17 |
| 14 | 13 | 20 | 8 | 1 | 2 | 1 | 2 | 13 | 17 | 8 | 19 | 9 | 15 | 14 |
| 13 | 21 | 6 | 11 | 8 | 10 | 16 | 1 | 2 | 13 | 13 | 22 | 12 | 13 | 14 |
| 8 | 19 | 13 | 10 | 1 | 17 | 1 | 8 | 9 | 10 | 4 | 12 | 17 | 11 | 13 |
| 15 | 9 | 14 | 15 | 6 | 14 | 1 | 2 | 13 | 14 | 19 | 9 | 14 | 13 | 11 |
| 9 | 19 | 12 | 18 | 8 | 11 | 17 | 1 | 13 | 21 | 11 | 8 | 12 | 2 | 13 |
| 14 | 15 | | | | | | | | | | | | | |

Appendix 17 - Substitution cipher model

## [Appendix F] Z340 Cipher investigation results

| Time | Pop Size | Fitness | Selection | Mutation | Crossover | Max. Final Fitness | Change in max fitness | Change in Mean Fitness | No. of Generations |
|---|---|---|---|---|---|---|---|---|---|
| 30 | 100 | N-gram | Roulette | Multi | Dual | 88.95457 | 0 | -46.0523 | 502 |
| 30 | 100 | Reduced | Roulette | Multi | Single | 85.19211 | 0 | -28.459 | 14731 |
| 30 | 100 | Reduced | Stochastic | Multi | Dual | 83.99485 | 1.714214 | -44.7712 | 15790 |
| 60 | 300 | N-gram | Roulette | Multi | Dual | 90.62201 | 0.942328 | -38.8652 | 341 |
| 60 | 300 | Reduced | Roulette | Multi | Single | 91.12437 | 0 | -39.8014 | 10349 |
| 60 | 300 | Reduced | Stochastic | Multi | Dual | 86.00219 | 0 | -45.5751 | 11223 |
| 60 | 300 | N-gram | Roulette | Multi | Dual | 92.20771 | 0 | -54.1404 | 338 |
| 60 | 300 | Reduced | Roulette | Multi | Single | 83.3639 | 0.0648 | -55.0258 | 10408 |
| 60 | 300 | Reduced | Stochastic | Multi | Dual | 83.32359 | 0 | -58.9867 | 11284 |
| 60 | 300 | N-gram | Roulette | Multi | Dual | 92.20771 | 0 | -54.1404 | 338 |
| 60 | 300 | Reduced | Roulette | Multi | Single | 83.3639 | 0.0648 | -55.0258 | 10408 |
| 60 | 300 | Reduced | Stochastic | Multi | Dual | 83.32359 | 0 | -58.9867 | 11284 |
| 60 | 300 | N-gram | Roulette | Multi | Dual | 91.7573 | 0 | -57.9596 | 329 |
| 60 | 300 | Reduced | Roulette | Multi | Single | 89.07682 | 1.858945 | -58.2668 | 9961 |
| 60 | 300 | Reduced | Stochastic | Multi | Dual | 86.9459 | 0 | -65.6666 | 10785 |
| 60 | 300 | N-gram | Roulette | Multi | Dual | 94.78381 | 0 | -59.9092 | 338 |
| 60 | 300 | Reduced | Roulette | Multi | Single | 85.03836 | 0.356766 | -57.213 | 10370 |
| 60 | 300 | Reduced | Stochastic | Multi | Dual | 87.72695 | 0 | -65.9269 | 11125 |
| 60 | 300 | N-gram | Roulette | Multi | Dual | 91.92987 | 0 | -55.6394 | 322 |
| 60 | 300 | Reduced | Roulette | Multi | Single | 85.83101 | 0.585522 | -59.6427 | 10128 |
| 60 | 300 | Reduced | Stochastic | Multi | Dual | 85.0641 | 0 | -64.9821 | 10711 |
| 60 | 300 | N-gram | Roulette | Multi | Dual | 91.7451 | 0 | -57.0528 | 326 |
| 60 | 300 | Reduced | Roulette | Multi | Single | 83.0986 | 0 | -55.1158 | 9982 |
| 60 | 300 | Reduced | Stochastic | Multi | Dual | 90.40165 | 0 | -69.7312 | 10803 |

Appendix 18 - Z340 investigation results