UNIVERSIDAD DE GRANADA

TRABAJO FIN DE MASTER

INGENIERÍA INFORMÁTICA

Cryptanalysis Ciphertext Based Genetic Algorithms

Breaking Transposition and Substitution Ciphers with Genetic Algorithm

Autor

Abdullah Taher Saadoon AL Muswai (alumno)

Directores

Juan Julián Merelo Guervós (tutor)

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Granada, septiembre de 2020

Cryptanalysis Ciphertext Based Genetic Algorithms

Breaking Transposition and Substitution Ciphers with Genetic Algorithm.

Autor
Abdullah Taher Saadoon AL Muswai (alumno)

Directores
Juan Julián Merelo Guervós (tutor)

Cryptanalysis Ciphertext Based Genetic Algorithms: Breaking Transposition and Substitution Ciphers with Genetic Algorithm

Abdullah, AL Musawi(student)

Keywords: Cryptanalysis, Genetic Algorithm, Transposition Cipher, substitution cipher, decryption ,optimization search.

Abstract

This thesis describes a method of deciphering messages encrypted with transposition cipher and substitution cipher utilising a Genetic Algorithm to search the keyspace. There are many tools used in the cryptanalysis, Genetic algorithm(GA) is an optimization search tool to find the best solution.

In this project, A

fitness measure based on **English Letter Frequencies** for random of text is described which got it by generating random popultaion. The results are compared to those given using a previously published technique and found to be superior.

Yo, **Abdullah Taher Saadoon AL Muswai**, alumno de la titulación máster de Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI A11518169, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Master en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Abdullah Taher Saadoon AL Muswai

Granada a 10 de septimpre de 2020.

D. **Juan Julián Merelo Guervos (tutor)**, Profesor del Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Cryptanalysis Ciphertext Based Genetic Algorithms,Breaking Transposition and Substitution Ciphers with Genetic Algorithm***, ha sido realizado bajo su supervisión por **Abdullah Taher Saadoon AL Muswai (alumno)**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de mes de 201 .

Los directores:

Juan Julián Merelo Guervos (tutor)

Agradecimientos

Poner aquí agradecimientos...

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introduction

Cryptanalysis is the technique of deriving the original message from the cipher text without any prior knowledge of secret key or the erivation of key from the cipher text. A general technique for cryptanalysis, applied to all cryptographic algos is to try all the possible keys until the correct key is matched, it is known as exhaustive key search.

With every passing day, the computing ability of hardware is increasing manifold; therefore it becomes necessary to use long keys for avoiding exhaustive key search. All the other attacks applied to stream ciphers are compared to exhaustive key search in terms of data and memory complexity and if its complexity is less than exhaustive key search, then only these are considered as successful.

A symmetric key cipher, especially a stream cipher is assumed secure, if the computational capability required for breaking the cipher by best-known attack is greater than or equal to exhaustive key search. Cryptanalysis is the science of making encrypted data unencrypted use convert cipher text to plaintext because cryptanalysis used to convert plaintext to cipher text and used cryptanalysis Return to plaintext or clear text or original text cryptanalysis is used to break codes by finding weaknesses. There are many techniques used in the cryptanalysis. This project used the genetics algorithm [1].

The genetic algorithm is a search algorithm based on the mechanics of natural selection and natural genetics. The genetic algorithm belongs to the family of evolutionary algorithms, along with genetic programming, Evolution strategies and evolutionary programming. The set of operators usually consists of mutation, crossover and selection [2].

A genetic algorithm has proven to be reliable and powerful optimization technique in a wide variety of applications. It can be applied to both texts and images. Genetic algorithm is secure since it does not utilize the natural numbers directly. The genetic algorithm used for generating keys that it should be good in terms of coefficient of autocorrelation [3].

Apply the technique of genetic algorithms to the problem of finding the key to a particular Transposition Cipher. Since Genetic Algorithms are primarily used to efficiently search a large problem space we thought they would be ideally suited for searching the large key space [4].

1.1. aim of the project

The main purpose of the project is examine the possible applications of genetic algorithms in cryptology,with the emphasis of the research being in the application of a genetic algorithm in the which explores the plaintext from cipher text based on genetic algorithm (GA) which is used for suggesting decryption key. The genetic algorithm (GA) is a search tool to insure high probability of finding a solution by decreasing the amount of time in the key space searching. this project covers two ciphers type: Transposition Cipher and Substitution Cipher. We present a review of Transposition Cipher ,Substitution Cipher, English Letter Frequencies and genetic algorithm in chapters ***** which provides enough background to understand the techniques applied and to assess the usefulness of the results obtained.

1.2. project organization

In additional to chapter one (the introduction), this project is organized into three other chapters: -Chapter * **"Theoretical Background"** this chapter introduces terms, definitions, and descriptions that are used frequently throughout this chapter. These terms focus on Cryptanalysis, Transposition Cipher, and description Genetic algorithm (GA). Chapter Three **"Design and Implementation: Cryptanalysis based on Genetic Algorithm"** this chapter shows the design of the project and explains the algorithms that support this work. Chapter Four **Çonclusions and Future Work"** this chapter concludes this project and finding the outline future works in the area of Cryptanalysis.

Capítulo 2

Planning

This chapter presents the work plan to follow to carry out this study. First of all, the requirements necessary to achieve the proposed objective will be captured, and then the work planning will be detailed, with an estimate of both work and material costs as well as a distribution of time between tasks.

2.1. Investigation stages

The steps of the research process of this study are as follows:

1. **Initiate a preliminary investigation of the Cryptanalysis:** review many of theses and papers looking for information about Cryptanalysis, check what evolution they have followed over the years and how the Cryptanalysis systems have been developed.

2. **delve into the Transposition and Substitution ciphers:** investigate further in these types that are really going to focus on the study

3. **Select the algorithm:** investigated in the previous steps, deciding the most appropriate option.

4. **delve into genetic algorithm:** Study and understand all stages of genetic algorithm evolution and how to solve current problems, review the papers that related with GA.

5. **Implement genetic algorithm:** create a set of classes, each of them performs stage of GA stages and all of them has a set of methods to divide its stage to small tasks every task solves a different problem.

6. **create a new proposal for the study:** for the objective of the study, new proposals that can compete with the state of the art must be proposed.

7. **Implement my new proposal:** that will be part of the experiment have been decided, they must be implemented in the same way as the state of the art algorithm. Check of new that all the functionality of the experiment is valid with the new implementation.

8. **Obtain the results of the experiment:** obtain through the implementation of the experiment, can visualize the data in tables and graphs that are suitable for the study.

9. **analysis of the results obtained:** compare the results of each method with the *baseline* reference method and with the state of the art method, as well as between them, to discover which method behaves best for each case.

10. **Consider future work to be done:** assess the aspects it is possible to continue progressing in the methods proposed in the study, propose new tasks or challenges within this field of research, that have not been covered in this study.

2.2. Work plan

This section is subdivided into cost estimation and time estimation. In the first place, The cost of the infrastructure used is evaluated, as long as the material that has been used is available for free and free. From there we can make a base budget from which to start. The time estimate should be realistic and meet the requirements within a competent time frame so that you do not need to postpone times at the last minute due to poor planning. If so, it would be necessary to bear the consequent increase in costs.

2.2.1. Cost estimate of materials and infrastructure

First of all, the computer with which this study has been developed will be taken, the following hardware and software have been used for this study: **Computer Type:**Lenovo IdeaPad Z510. **Processor:**Intel(R) Core(TM) i7-4702MQ CPU@ 2.20GHz 2.20GHz. **Installed memory (RAM):** 8.00 GB. **graphics card:** Intel(R) HD Graphics 4600, Memory:2176 MB. **Hard Disk:** HDD 1TB

Operating system: Win10. **Programming language:** java. **Environment:** NetBeans 8.1. All the research, implementation, and evaluation

tasks have been carried out with this team: documentation, review of the literature, writing of the report, development, implementation of the algorithms, and execution of the experiment to obtain results.

information and documentation required for the work are available at the University of Granada Thanks to the agreements that the University establishes with some documentary databases such as Scopus, it has been possible to access a multitude of papers and theses on the subject to be investigated. Furthermore, the Google scholer has been very useful to access other papers not found in Scopus.

The development of the work has been carried out entirely on the Win10 OS, The main software tools used have been open source so they have not entailed additional cost, Netbeans as the main IDE for development and writing the report of the project in LaTeX using the TeXstudio, available for win 10, and we used Visual Studio Code as an IDE to deal with the repository of the project that has hosted on **GitHub** [5] platform, and we used **Travis CI** [6] to host continuous integration service used to build and test software projects hosted at GitHub,Travis CI provides various paid plan for private projects, and a free plan for open source. Of all this material exposed in the previous paragraphs, the work has only required buying the personal laptop and the broadband network available, The rest of the infrastructure and materials have either been free software products or have been contributed by the University of Granada.

2.2.2. Time distribution to the tasks

we used the concepts of iterations and sprints to distribute the work throughout the weeks, in which a series of tasks have been developed that they fulfill a specific objective.

The Iteration and Sprint Planning meeting is for team members to plan and agree on the stories or backlog items they are confident they can complete during the sprint and identify the detailed tasks and tests for delivery and acceptance

The planning that has been foreseen for the development of the project is as follows:

Weeks	10
Estimated total time (h)	240h
date of Starting Point	01-06-2020
date of end Point	10-09-2020

2.2.3. Preparatory phase

In this phase, the development of the project will begin, which has an estimated effort of 40 hours. A feasibility study of the project idea will be carried out, an investigation about the possible tools to use

Iteration 1: Analysis and study of the project.

week	1
Total expected time (h)	40h
date of Starting Point	01-06-2020
date of end Point	06-06-2020

Sprint 1: Description of the Project

Product	Description	Week	Time
Objectives	Description of the objectives.	1	1.5
functionalities	Description of functionalities.	1	3
Motivations	Personal motivations around the project.	1	1.5
Programming	Description of possible programming methods for the proposed project.	1	4

Sprint 2: Research and preparation of the environment.

Product	Description	Week	Time(h)
Study and analysis of market	Search for similar projects and study on their components	1	10
Technologies and tools	Study of technologies and tools necessary for the development of the project	1	15
Work environment	Creation of the GitHub repository and development environment of the project	1	8

2.2.4. Implementation and testing phase

the objective of this phase is getting an general idea of this project and at the end of this phase, a software-implemented incrementally ha-

ve been obtained, where each module has been tested until integration and validation. A total of 280 hours has been planned for this phase, and it has been distributed in the following iterations:

Iteration 2: Implementation of the main classes.

The objective of this iteration is the implementation of the main classes which will create the system,For this operation, the following schedule has been established:

week	6
Total expected time (h)	120h
date of Starting Point	07-06-2020
date of end Point	18-07-2020

Product	Description	Week	Time(h)
information	Study needed: java libraries, fitness measure, Crossover operators ….	6	10
Population class	Populate the initial population with completely random solutions	6	10
Transposition and Substitution classes	Try to decode the cipher texts.	6	25
Fitness class	Calculate the fitness equation which is using English letter frequencies	6	20
Crossover class	Apply Crossover operators on population.	6	30
Mutation class	Apply Mutation operator on population.	6	5
Testing	Implementation of the tests unit of the classes developed in this iteration.	6	20

Iteration 3: Experimental results and Improvement. The objecti-

ve of this iteration is trying to get a result from the classes then try to improve all of them to get better results,For this iteration, the following schedule has been established:

week	10
Total expected time (h)	120h
date of Starting Point	18-07-2020
date of end Point	18-08-2020

Product	Description	Week	Time(h)
information	Review papers that related for each class and compare the results with.	10	15
Population class	repopulate populations with different cases	10	10
Transposition and Substitution classes	Trying to improve these classes to enhance run time	10	25
Fitness class	Compare fitness results before and after enhancing and adding the tables of English letter frequencies	10	25
Crossover class	Try to find new crossover operators and compare the result with previous result.	10	25
Mutation class	Try to balance mutation ratio with each population.	10	10
Testing	Implementation of the tests unit of the classes developed in this iteration.	10	10

2.2.5. Documentation phase

After the project is completed and approved, it is time to document all of the steps and how to work, and document the achieving result.

the documentations have been divided into two parts. The first one corresponds to the documentation in the Github repository. This documentation has as objective to give an idea about the project in general then delve in details until arrives in programming steps that explain all of the class and their methods and variables, Anyone who accesses to the repository, he can see all this information as a free software project.

Other documents made are those for the project. In this case, more documentation focused on the scope of the project will be implemented: planning, research, development, Conclusions.

Iteration 4: GitHub Documentation. In this phase has an estimated effort of 50 hours.

week	11
Total expected time (h)	50h
date of Starting Point	18-08-2020
date of end Point	24-08-2020

Product	Description	Week	Time(h)
ABSTRACT and Introduction	Insert ABSTRACT and Introduction of the project in README File	11	5
Implementation	Add Implementation of all the part of the project.	11	15
Programming	Add description of all the class of the project and explain them step by step.	11	20
Experimental results	Put the Experimental results for each class in README	11	10

Iteration 5: Project documentation.

In this phase has an estimated effort of 150 hours.

week	13
Total expected time (h)	150h
date of Starting Point	24-08-2020
date of end Point	09-09-2020

Product	Description	Week	Time(h)
Information	Review documentation about Latex	13	5
Create LaTeX file	Generation of template, packages, .abstract and cover page	13	3
Introduction	Writ all part of introduction	13	10
Planning	Writ all part of the planning chapter	13	20
Chapter 3			
Chapter 4			
Chapter 5			
Chapter 6			
Chapter 7			

Capítulo 3

Theory Background

3.1. Introduction

Cryptanalysis is the technique of extracting useful information about the key by observing the plaintext and cipher text using cryptanalysis try to break the secrecy provided by the cipher. There is no fixed method for cryptanalysis and every cipher is a different challenge to the attacker and hence demands different insight to attack[7],The study of cipher text in an attempt to restore the message to plaintext is known as cryptanalysis. Cryptanalysis is equally mathematically challenging and complex as cryptography. Because of the complexity involved with cryptanalysis work this document is only focused on the basic techniques needed to decipher monoalphabetic encryption ciphers and cryptograms[8]. In this chapter, explained the history of cryptanalysis, the technology of cryptanalysis, transposition cipher, substitution cipher and description genetics algorithm (GA).

3.2. Cryptanalysis

Cryptanalysis is the technique of deriving the original message from the ciphertext without any prior knowledge of secret key or derivation of key from the ciphertext. A general technique for cryptanalysis, applicable to all cryptographic algorithms is to try all the possible keys until the correct key is matched, it is known as exhaustive key search. With every passing day, the computing ability of hardware is increasing manifold; therefore it becomes necessary to use long keys for avoiding exhaustive key search [1] and till today, many cryptanalytic attacks are developed based on these. Each variant of these have different methods

to find distinguisher and based on the distinguisher [9].

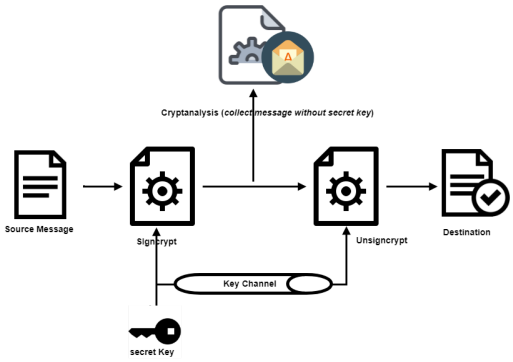

Figura 3.1: Cryptanalysis cipher text

3.2.1. CLASSIFICATION OF ATTACKS

The main goal of a cryptanalyst is to obtain maximum information about the plaintext (original data).Classification of attacks can be done on following basis [10]:

1. **Cipher text only attacks:** This is the most powerful attack. The attacker has only the knowledge of cipher text. This type of attack is successful only on the weakest of the ciphers.

2. **Known plaintext attack:**attacker has the knowledge of plaintext and the corresponding cipher text, e.g. if an attacker is eavesdropping then he can also guess the plaintext corresponding to some cipher texts depending upon the position or state of communication, in other words, In this type a cryptanalyst have plaintext and their corresponding cipher text . Attacker tries to find out the relation between these two.

3. **Chosen plaintext:**the attacker can choose its plaintext and get the cipher text corresponding to those chosen cipher text or The attacker obtain the various ciphertext corresponding to an arbitrary set of plaintext.

4. **Chosen cipher text:** attacker is able to get the decrypted plaintext corresponding to his choice of cipher text. This attack is same as the chosen plaintext, but in a reverse direction which means The attacker obtain the various plaintext corresponding to an arbitrary set of cipher text

5. **Adaptive chosen plaintext:**attacker first observes a large number of cipher texts. Based on the distribution of the cipher texts the attacker chooses a plaintext to get the corresponding cipher text which means the attacker chooses subsequent set of plaintext which is based on the information obtain from previous encryption methods.

6. **Related Key:** This is a relatively new attack model. Here the attacker can encrypt two plaintext (same plaintext or the two plaintexts with a constant difference) with two keys, which have a fixed relation Between each other. This attack model is very weak as there is very little chance for the attacker to get encryption with two keys with a Constant relation. For lightweight block ciphers as the key is written to the device, this type of attack is not very probable.

3.2.2. Cryptanalytic technique

In this section we will explain various cryptanalytic technique. As said earlier that, there are no fixed methods for cryptanalytic techniques for any block ciphers. But there are some methods which can be applied to every ciphers with some variation, though there can not be any guarantee that these methods may break the cipher. Cipher designers apply these methods to analyze security level for the computational security. Informally, broadly classify these techniques as brute force techniques non-brute force techniques. As the name suggest brute force techniques involves search of entire key space. Other techniques utilize the weakness in the structure of the ciphers to find key bits [7].

Brute force technique:

A brute-force attack is a can be used to attempt to decrypt any encrypted data. Such an attack might be used when it is not possible to take advantage of other weaknesses in an encryption system (if any exist) that would make the task easier. When password guessing, this method is very fast when used to check all short passwords, but for longer passwords other methods such as the dictionary attack are used because a brute-force search takes too long. Longer passwords, passphrases and keys have more possible values, making them exponentially more difficult to crack than shorter ones. can be made less effective by obfuscating the data to be encoded making it more difficult for an attacker to recognize when the code has been cracked or by making the attacker do more work to test each guess [7].

Non-brute force techniques:

In a non-brute-force attack, a single (usually common) password is tested against multiple usernames or encrypted files. The process may be repeated for a select few passwords. In such a strategy, the attacker is Generally not targeting a specific user. Non brute-force attacks can be mitigated by establishing a password policy that disallows common passwords [7].

3.3. CipherText

Ciphertext is encrypted text transformed from plaintext using an encryption algorithm. Ciphertext can't be read until it has been converted into plaintext (decrypted) with a key. The decryption cipher is an algorithm that transforms the ciphertext back into plaintext. The term cipher is sometimes used as a synonym for ciphertext. However, it refers to the method of encryption rather than the result.

3.3.1. Types of ciphers

There are various types of ciphers but We are focusing on only two of them that related to the work done , including:

Transposition Cipher

The transposition cipher is rearranged (change position only) the characters in the message but not change the characters. Transposition cipher have a pool of keys and ciphertext that rearranged the ciphertext for M times depended on the pool of keys. The output of transposition cipher saved in array of M locations we can called it *plaintextArray*. A simple transposition or permutation cipher works by breaking a message into fixed size blocks, and then permuting the characters within each block according to a fixed permutation, say P. The key to the transposition cipher is simply the permutation P. So, the transposition cipher has the property that the encrypted message contains all the characters that were in the plaintext message. In other words, the unigram statistics for the message are unchanged by the encryption process. The size of the permutation is known as the period. Let's consider an example of a transposition cipher with a period of ten 10, and a key P=7,10,4,2,8,1,5,9,6,3. In this case, the message is broken into blocks of ten characters, and after encryption the seventh character in the block will be moved to position 1, the tenth moved character in the block will be moved to position 2, the forth is moved to position 3, the second to position 4, the eighth to position 5, the first to position 6, the fifth to the position 7, the ninth to the position 8, the sixth to the position 9 and the third to position 10.

In Table 3.1 shows the key and the encryption process of the previously described transposition cipher. It can be noticed that the random string "X"was appended to the end of the message to enforce a message length, which is a multiple of the block size.It is also clear

that thedecryption can be achieved by following the same process as encryption using the ïnverse.ºf the encryption permutation. In this case the decryption key, P-1 is equal to 6,4,10,3,7,9,1,5, 8,2.

KEY:
Plaintext: 1 2 3 4 5 6 7 8 9 ***10*** Ciphertext:7 ***10*** 4 2 8 1 5 9 6 3
ENCRYPTION:
Position : 123456789***10*** 1234 5678 9*10* 123456789***10*** Plaintext : TRANSPOSITION _ ALGORITHMXXXXXXX Ciphertext OTNRSTSIPAGI _ OOIARLNXXXHXTXXXM

Cuadro 3.1: Transposition Cipher Example

Substitution Cipher

A cipher is an algorithm for encrypting plain text into cipher text and vice versa. The substitution cipher replaces every instance of a particular letter in the plain text with a different letter from the cipher text. Thus a substitution cipher key can be defined as the set of one-to-one mappings relating every letter in the plain text alphabet with the corresponding letter in cipher text alphabet. Such a key is normally defined using a table 3.2 and an example key is included below.

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

Cuadro 3.2: Substitution Cipher Example

Thus the ***HERE THE UNIVERSITY OF GRANADA*** can be encrypted to ***QVOV PQV WGIDVOJPL QA ROEGEXE***,On the surface this cipher seems to be a strong one since there are 26 possibilities to choose from for the first letter, 25 for the second, 24 for the third and so on. One can clearly see that there are in fact 26! possible keys.

$$26! = 26 * 25 * 24 * \ldots * 1 = 4.03^{26} \tag{3.1}$$

However this cipher is particularly vulnerable to a technique known as frequency analysis since although it does change the letters in the plain text to different ones in the cipher text it does not change the underlying frequency of those letters. Thus by comparing the frequencies of letters in the cipher text to a table of known letter frequencies for the plain text language the key space can be reduced drastically. Furthermore you can also group letters into n-grams where n represents the number of letters in the n-gram and look up the corresponding frequencies for those [4].

3.4. Genetic Algorithm

Nature has always been a great source of inspiration to all mankind. Genetic Algorithms (GAs) are search based algorithms based on the concepts of natural selection and genetics. GAs are a subset of a much larger branch of computation known as Evolutionary Computation[11]. GAs were developed by John Holland and his students and colleagues at the University of Michigan, most notably David E. Goldberg and has since been tried on various optimization problems with a high degree of success[11].

The genetic algorithm is a general method of solving problems to which no satisfactory, obvious, solution exists. It is based on the idea of emulating the evolution of a species in nature, so the various components of the Algorithm are roughly analogous to aspects of natural evolution. Common mathematical tasks amenable to genetic solutions include computing a curve to fit a set of data. Often these operators consist of flipping a single. Random bit of one individual or swapping two randomly selected substrings from a pair of parents to generate a new child. To simulate Darwinian survival of the fittest some representation of the fitness of the individuals must be generated [12].

Genetic Algorithm Attack is more complicated than the any other attack. This is because a pool of solutions is being maintained, rather than a single solution. An extra level of complexity is also present because of the need for a mating function and for a mutation function [13].

Genetic Algorithms are sufficiently randomized in nature, but they perform much better than random local search (in which we just try various random solutions, keeping track of the best so far), as they exploit historical information as well [11].

3.4.1. The steps of the genetic algorithm

The genetic Algorithm has six steps which can evolve solutions to the search problem the following [14]:

Initialization(Population)

The initial population of candidate solutions is usually generated randomly across the search space. However, domain-specific knowledge or other information can be easily incorporated in the generation of the initial population. The population is usually defined as a two dimensional array of size population, size x , chromosome size. There are two primary methods to initialize a population in a GA. They are:

Random Initialization:

Populate the initial population with completely random solutions.

Heuristic initialization:

Populate the initial population using a known heuristic for the problem

Evaluation(Fitness):

Once the population is initialized, or an offspring population is created, the fitness values of the candidate solutions are evaluated. A fitness function should possess the following characteristics:

1. The fitness function should be sufficiently fast to compute.
2. It must quantitatively measure how fit a given solution is or how fit individuals can be produced from the given solution.

Selection:

Selection allocates more copies to solutions with better fitness values and thus imposes the survival-of-the-fittest mechanism on the candidate solutions. The main idea of selection is to prefer better solutions to worse ones, and many selection procedures have been proposed to accomplish this idea,

Recombination(Crossover):

Recombination combines bits and pieces of two or more parental solutions to create new, possibly better solutions (i.e. offspring). There are many ways of accomplishing this, and achieving competent performance depends on getting the recombination mechanism designed properly; but the primary idea to keep in mind is that the offspring under recombination will not be identical to any particular parent and will instead combine parental traits in a novel manner [15].

In GAs a crossover can be of following operators:

Single Point Crossover:

In this one-point crossover, a random crossover point is selected and the tails of its two parents are swapped to get new off-springs, as shown in figure 3.2.

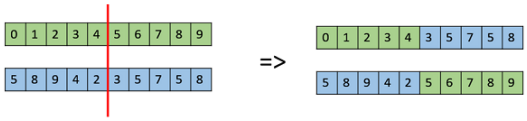

Figura 3.2: One Point Crossover operator

Multi Point Crossover:

Multi point crossover is a generalization of the one-point crossover wherein alternating segments are swapped to get new off-springs, as shown in figure 3.3

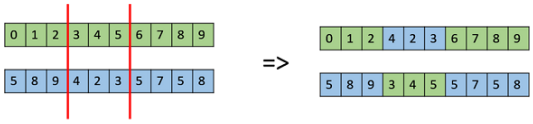

Figura 3.3: Multi Point Crossover operator

Uniform Crossover:

In a uniform crossover, we don't divide the chromosome into segments, rather we treat each gene separately. In this, we essentially flip a coin for each chromosome to decide whether or not it'll be included in the off-spring. We can also bias the coin to one parent, to have more genetic material in the child from that parent, as shown in figure **??**.

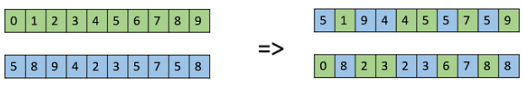

Figura 3.4: Uniform Crossover operator

Mutation:

While recombination operates on two or more parental chromosomes, mutation, locally but randomly, modifies a solution. Again, there are many variations of mutation, but it usually involves one or more changes that are made to an individual's trait or traits. In other words, mutation performs a random walk in the vicinity of a candidate solution, as shown in figure 3.5.

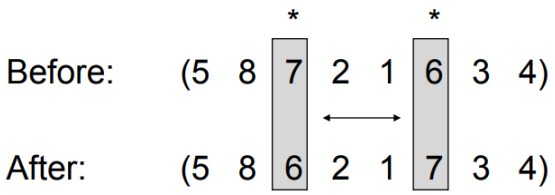

Figura 3.5: Mutation Example

Replacement:

The offspring population created by selection, recombination, and mutation replaces the original parental population. Many replacement techniques such as elitist replacement, generationwise replacement and steady-state replacement methods are used in GAs.

3.4.2. Advantages and Limitations of Genetic Algorithms

Advantages

GAs have various advantages which have made them immensely popular. These include:

1. Does not require any derivative information (which may not be available for many real-world problems).
2. Is faster and more efficient as compared to the traditional methods.
3. Has very good parallel capabilities.
4. Optimizes both continuous and discrete functions and also multi-objective problems.
5. Provides a list of "good" solutions and not just a single solution.

6. Always gets an answer to the problem, which gets better over the time.

7. Useful when the search space is very large and there are a large number of parameters involved.

Limitations

Like any technique, GAs also suffer from a few limitations. These include:

1. GAs are not suited for all problems, especially problems which are simple and for which derivative information is available.

2. Fitness value is calculated repeatedly which might be computationally expensive for some problems.

3. Being stochastic, there are no guarantees on the optimality or the quality of the solution.

4. If not implemented properly, the GA may not converge to the optimal solution.

3.4.3. Application areas of genetic algorithm (GA):

Genetic Algorithms are primarily used in optimization problems of various kinds, but they are frequently used in other application areas as well. Some of the areas in which Genetic Algorithms are frequently used explained in the following:

Optimization:

Genetic Algorithms are most commonly used in optimization problems wherein we have to maximize or minimize. Given objective function value under a given set of constraints. The approach to solve Optimization problems has been highlighted throughout the tutorial.

Economics:

GAs are also used to characterize various economic models like the cobweb model, game theory equilibrium resolution, asset pricing, etc.

Parallelization:

GAs also have very good parallel capabilities, and prove to be very effective means in solving certain problems, and provide a good area for research.

Image Processing:

GAs are used for various digital image processing (DIP) tasks as well like dense pixel matching.

Vehicle routing problems:

With multiple soft time windows, multiple depots and a heterogeneous fleet.

Scheduling applications:

GAs are used to solve various scheduling problems as well, particularly the time tabling problem.

Machine Learning:

genetics based machine learning (GBML) is a niche area in machine learning

Robot Trajectory Generation:

GAs have been used to plan the path which a robot arm takes by moving from one point to another.

Parametric Design of Aircraft:

GAs have been used to design aircrafts by varying the parameters and evolving better solutions.

Capítulo 4

Design and Implementation of Proposed Work

This chapter will show the architectural, Algorithm, and class diagrams of the application,In this chapter also, the implementation of the proposed work will be explained which shows how genetic algorithm approach can be applied to cryptanalysis task, which does not have clear solution except for exhaustive search: breaking the Transposition and substitution cipher. In this work, cryptanalysis used GA for searching the decryption key or getting maximum information about the decryption key. The Genetic Algorithm is a search and optimization techniques based On Darwin is the Principle of Natural Selection, In Figure 4.1 Flowchart of proposed method of cryptanalysis.

4.1. Design of the work

As we saw in Figure 4.1 that we have nine main parts in the project Consequently, project design will content 9 dependency classes, first of them will be the main class which controls other classes as shown in Figure 4.2 and explains inheritance relationship between classes, In this section, we will explain the stages of project development as follows:

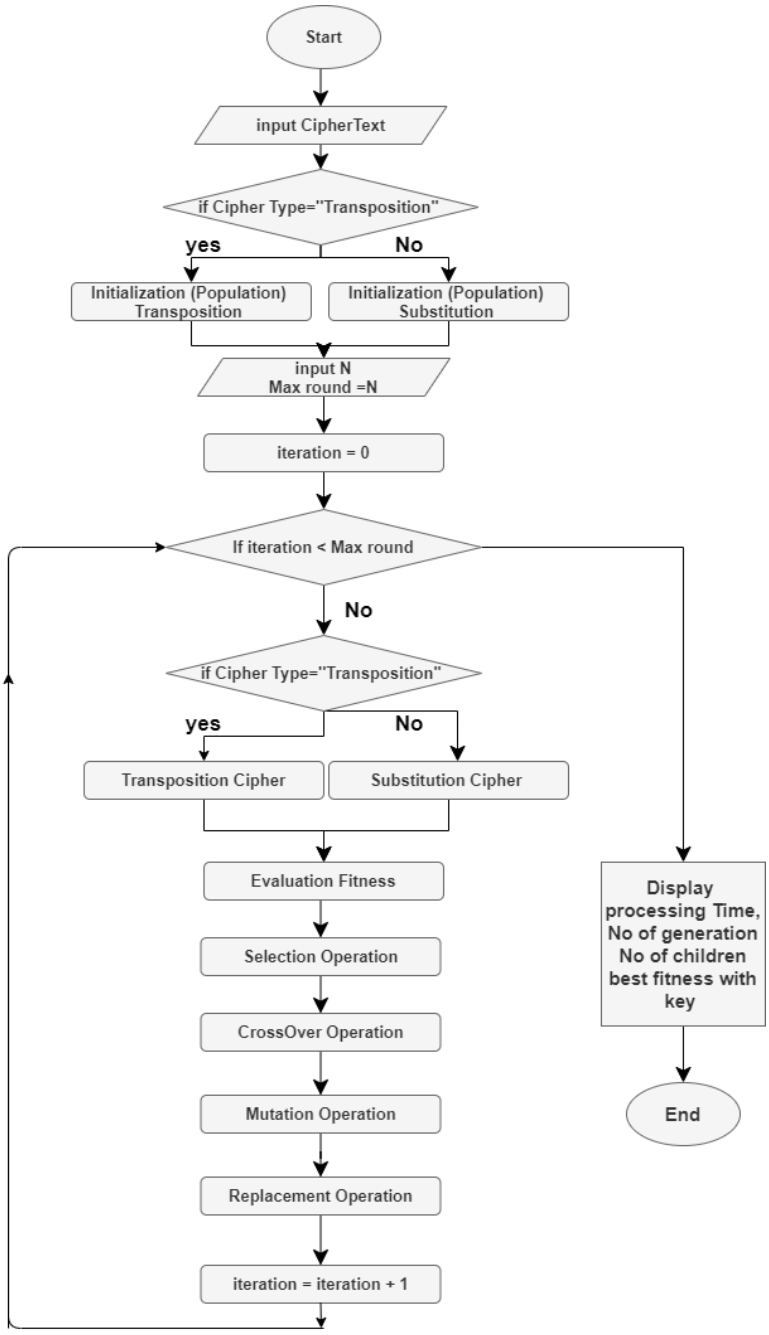

Figura 4.1: Flowchart Of Proposed Method Of Cryptanalysis

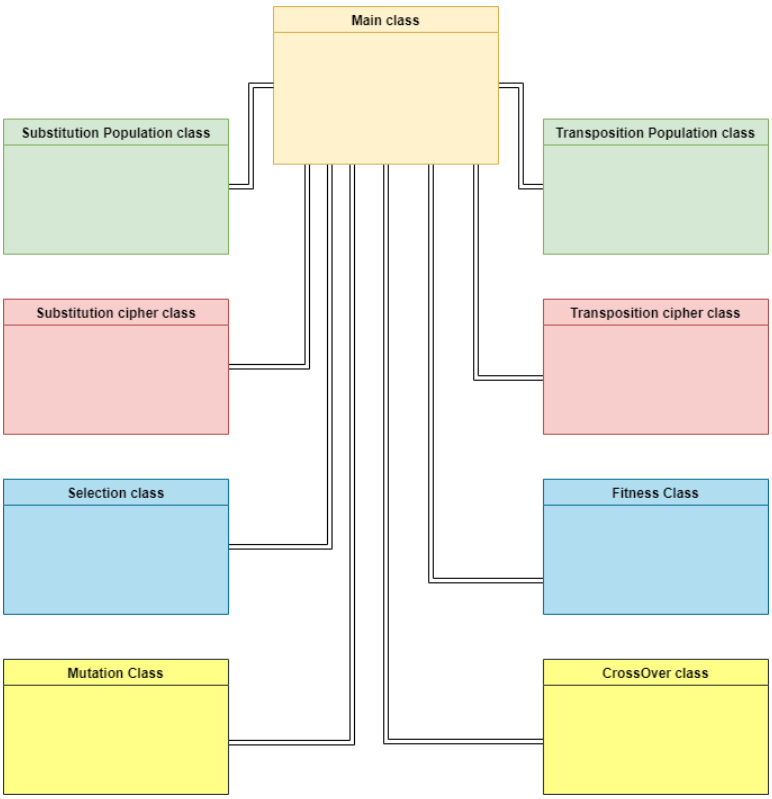

Figura 4.2: Inheritance relationship between the classes

4.1.1. Population

This part of the project is divided into two parts, depending on the cipher type:

Transposition cipher population:

In the initialization stage, generate a pool of random keys (size population), when the length of the key is N digits (chromosome size) and the pool size is M keys(size population). Also, The key condition is random and non-repetitive in each key.

Substitution cipher population:

on the other side, generate a pool of keys (size population), when the length of the key is 26 characters which equals the number of English

letters (chromosome size) and the pool size is M keys(size population). Also, The key condition is random and non-repetitive in each key. These keys are changeable by the other stages of GA and the better one used in derivation the plaintext.

4.1.2. Breaking Cipher

in this stage, we will try to break the ciphertext using the keys which we got from the previous stage.

Transposition Cipher:

The transposition cipher is rearranged (change position only) the characters in the message but not change the characters. Transposition cipher has a pool of keys and ciphertext that rearranged the ciphertext for M times depending on the pool of keys. The output of transposition cipher saved in an array of M locations we can be called *plaintext array.*

Substitution Cipher:

The substitution cipher replaces every instance of a particular letter in the ciphertext with a different letter from the cipher key, Thus a substitution cipher key can be defined as the set of one-to-one mappings relating every letter in the ciphertext alphabet with the corresponding letter in plaintext alphabet, so The Substitution cipher is changing the characters in the message but not change the characters positions, substitution cipher has a pool of keys and ciphertext that changed the ciphertext for M times depending on the pool of keys. The output of substitution cipher saved in an array of M locations which we can be called *plaintext array.*

4.1.3. Evaluation Fitness

Fitness is evaluated based on the unigrams(one letter) frequencies, the bigrams (sequence of two letters) frequencies in the decrypted ciphertext, and Trigrams (sequence of three letters) frequencies in the decrypted ciphertext. the tables below illustrated the most popular bigrams and Trigrams in the English language. Trigrams and diagrams are computationally expensive the fitness calculation. The idea of the fitness function is the following [16]:

1. We take large text corpus and calculate the occurrences of unigrams, bigrams, and trigrams

$$C_i^u, C_{ij}^b, C_{ijk}^t$$

2. Then we sum all counters to get total sum of unigrams, bigrams and trigrams,

$$S_u, S_b, S_t$$

3. Then we calculate reference frequencies of each unigram, bigram and trigram,

$$R_i^u = \frac{C_i^u}{S_u}, R_{ij}^b = \frac{C_{ij}^b}{S_b}, R_{ijk}^b = \frac{C_{ijk}^t}{S_t}$$

4. Then, using the same method, we calculate the frequencies within the target text to obtain partial frequencies of each unigram, bigram and trigram,

$$P_i^u, P_{ij}^b, P_{ijk}^t$$

.

5. Then we calculate the fitness score using the following formula:

$$F_i = \alpha \sum (R_i^u - P_i^u) + \beta \sum (R_{ij}^b - P_{ij}^b) + \gamma \sum (R_{ijk}^t - P_{ijk}^t)$$

where alpha, betha and gamma are the weights we assign to the importance of unigrams, bigrams and trigrams respectively. This implementation uses 1/6, 1/3 and 1/2, assigning most of the weight to trigrams

frequencies of unigrams, bigrams and trigrams are shown in the tables 4.14.24.3 respectively.

A : 8.55	K : 0.81	U : 2.68
B : 1.60	L : 4.21	V : 1.06
C : 3.16	M : 2.53	W : 1.83
D : 3.87	N : 7.17	X : 0.19
E : 12.10	O : 7.47	Y : 1.72
F : 2.18	P : 2.07	Z : 0.11
G : 2.09	Q : 0.10	
H : 4.96	R : 6.33	
I : 7.33	S : 6.73	
J : 0.22	T : 8.94	

Cuadro 4.1: English single letter frequencies

TH : 2.71	EN : 1.13	NG : 0.89
HE : 2.33	AT : 1.12	AL : 0.88
IN : 2.03	ED : 1.08	IT : 0.88
ER : 1.78	ND : 1.07	AS : 0.87
AN : 1.61	TO : 1.07	IS : 0.86
RE : 1.41	OR : 1.06	HA : 0.83
ES : 1.32	EA : 1.00	ET : 0.76
ON : 1.32	TI : 0.99	SE : 0.73
ST : 1.25	AR : 0.98	OU : 0.72
NT : 1.17	TE : 0.98	OF : 0.71

Cuadro 4.2: The bigram frequencies

THE : 1.81	ERE : 0.31	HES : 0.24
AND : 0.73	TIO : 0.31	VER : 0.24
ING : 0.72	TER : 0.30	HIS : 0.24
ENT : 0.42	EST : 0.28	OFT : 0.22
ION : 0.42	ERS : 0.28	ITH : 0.21
HER : 0.36	ATI : 0.26	FTH : 0.21
FOR : 0.34	HAT : 0.26	STH : 0.21
THA : 0.33	ATE : 0.25	OTH : 0.21
NTH : 0.33	ALL : 0.25	RES : 0.21
INT : 0.32	ETH : 0.24	ONT : 0.20

Cuadro 4.3: The trigram frequencies

4.1.4. Selection

In this operation, selection (choosing) the best keys only. The best key which has a high value of fitness. In the proposed work select only M/2 keys that have high fitness. To perform the selection operation needed a sorting function to sort pool of keys from high fitness to low fitness.

Fitness Proportionate Selection is one of the most popular ways of parent selection. In this stage, every individual can become a parent with a probability that is proportional to its fitness. Therefore, fitter individuals have a higher chance of mating and propagating their features to the next generation. Therefore, such a selection strategy applies a selection pressure to the more fit individuals in the population, evolving better individuals over time.

4.1.5. CrossOver

from the selection steps we got the best half of the population after calculating the fitness values, in this section, we generate a new population from the parents to get the children which represent the next generation, we can generate 2 children by using (one point, two-point or uniform crossover operators)for each pair parents, but that gives us just half number of the original population, so can we use two methods to generate a new population, for instance, every 2 parents give us 4 children, the first 2 children are generated using one-point crossover operator and the other children are generated using uniform crossover operator, in another way to keep the same number of population, we can keep the parents to the next generation, Through experiences, we can discuss which one is best to use, for that we applied all of them in this part to make the application more flexible.

4.1.6. Mutation

In this operation, applying the mutation operation for the new population. To perform the mutation operation, two random numbers generated such as R1, and R2 representing two positions in each key then swap between the value of the position R1 and the value of the position R2. Repeat this operation for all keys in the "new population"pool.

4.1.7. Display The Results

In this operation, display the plaintext (clear text is unencrypted information) for storage or transmission after decrypting the ciphertext. Also, display the fitness values, the number of geneerations and te number individuals.

4.2. Implementation of the proposed work

4.2.1. Population Steps

the first class to start is Population.java in this class we created the Population which represents a set of chromosomes and each chromosome has the same length so, Population class has one constructor and many methods that follow: Population Constructor: this constructor has two parameters:

- NoOFkeys: No. of chromosomes
- lengthOFkey: length of the chromosome.

under this Constructor, we can call all of the methods to create our population and show it. The first method is *IFKEYEXIST* this method will generate a set of numbers between 1 and length of chromosome without repeat numbers (non-repetitive) and randomly, Method *check* is created to ensure there are no duplicated numbers and check method has two parameters the first is an array of numbers are saved before and the second is a new number which will save in the same array, and return true or false (if this new number exist in the array of the chromosome will return false in another case will return true and save it) after getting more then 2 chromosomes we have to check if there are duplicate chromosomes this step done with this method which called *checkrow* which receive all of the chromosomes that saved before and new chromosome after that will compare between them to find if there are duplicated between these chromosomes, then will return false if there are no chromosomes duplicated and save it, if not, it will repeat the step of creating new chromosome. Note* this population will use to breaking Transposition cipher when we wanted to create a population to breaking Substitution Cipher we have created a new population class with some changes as follow: the constructor of the substitution population class has just one parameter which is a no. of keys(No. of chromosomes) then we generated 26 characters from A to

z randomly without repeating, so the length of the chromosome will be 26 characters are sorted randomly and non-repetitive.

4.2.2. Trnsposition Steps

The second step is Transposition Cipher, in this class Transposition.java we will try to get the origin text by using exchange the positions of characters as I explained in the section of Transposition Cipher, so if we have 16 keys, we can get 16 texts that may be the plaintext, in this class, we have 4 methods and one constructor, this class has:

- Transposition constructor: this Constructor has two parameters the array of keys which we got from Population class and ciphertext, then we will call methods that follow:

  1. check if lenNotDivid We created this method to enforce the message length, adding an "X"to the end of the message, making the message proportional to the key length.
  2. change position this method splits the ciphertext as blocks, the length of each block equals the length of the key, then call the SortbyKey method with sending one of the keys and one of the blocks to exchange the characters positions in each block.
  3. SortByKey this method receives one of the keys and one block of ciphertext and then exchanges the positions of all characters in that block depending on that key.
  4. Print this method called in the Transposition constructor to print the array of PlainText after doing the Transposition.

4.2.3. Substitution Steps

Substitution.java we will try to get the origin text by using change the characters as I explained in the section of Substitution Cipher, so if we have 16 keys, we can get 16 texts that may be the plaintext, in this class, we created a constructor, under this constructor we created new array to save texts after doing Substitution process, then we called a substitution method which responsible to change ciphertext to another text depending on the keys, under this method, we convert each character in ciphertext to ASCII code then apply an equation to get the index of this character from the keys array, *an instance:*

key= K H N E I B F G M D L U J V S P C Y Q R T X Z O W A
ciphertex=HTSCRJ
then we can get every character ASCII
ASCII(H)=73
ASCII(H)= ASCII(H)-65 so, ASCII(H)=8,
GetIndex(8) form Key array
H=G
repeat this process to get every characters
so, PlainText= GUOQTM

4.2.4. Fitness Evaluation steps

with this step we created a new class, we called it, Fitness.java, this class contents on one constructor and set of methods to calculate fitness step by step, The first thing we did it is creating 6 arrays that follow:

- monogram: to save the unigram score(weight)
- Twochar:to save the bigram frequencies.
- TwoCharVale to save the bigram score(weight).
- Threechar:to save the trigram frequencies.
- ThreeCharVale to save the trigram score(weight).

now, we are going to start with Fitness constructor which receives the Array of PlainTexts which we got from Transipostion step and Array of keys(chromosomes), under this constructor, we initialized a set of arrays to save the summation of bigram frequencies of each PlainText, trigram frequencies summation, and unigram, and we created another array to save the final value of fitness for each PlainText, then we called methods to start calculating which follow:

- FitnessMethod: to save the unigram score(weight)
- fitnessequation:under this method, we will calculate the final value of fitness for each PlainText by using the equation and save it in Fitness array.

4.2.5. Selection Steps

After getting fitness array for each PlainText from the previous step, it is time to select the best parents to generate a new population from those parents, let us firstly sort the fitness array descending then select the best half of that array by using high fitness, in the Selection.java class, we have one selection contracture which receives three parameters (the keys Array(chromosomes), PlainTexts Array, fitness Array), under this contractor the first thing we have done it is initializing a set of arrays,

- selectkey: to save the best of keys.
- selectPlinText: to save the best plainText then we called two methods are the following:
- Sellsort:this method sorts the fitness array at the same time sorts the keys and plaintexts arrays depending on the fitness values. note: the sell sort is one of the best sort ways are used to sort set of numbers [17].
- Selection: this method saves the best half of keys and plaintexts in other arrays and preppers them To produce a new generation.
- Print: to print the best keys and plaintexts.

4.2.6. CrossOver Steps

in this class, we have to decide What better behavior to generate a new generation, for that, it's so important to step, as we explained in this section that there are many CrossOver operators to generate a new population dependent on the previous generation, we have used the 3 operators, before starting to create methods to perform these operators we had created a new class, we called it CrossOver class which contents one constructor to initialize set of arrays to save next generation and calls some methods that follow:

- crossing: this method receives the set of best parents whose we got them from the Selection step, this method takes each parent individually and splits it into two equal parts then keeps the first part without change and do ascend sorts to the second then marriages them to represent a new child *an example*

Parent 1 = 5 4 6 3 1 2 so, the Child 1= 5 4 6 1 2 3,
Parent 2 = 1 6 5 4 3 2 so, the Child 2= 6 1 5 2 3 4

- onePoint:this is another method to get a new population, but in this method, we will deal with two parents to generate Two children, as we explained in chapter two, so, this method takes two parents and keeps the first part of the first parent splits them into two equal parts for generating a first-child, keeps the first part of the first parent without change and copy the remaining unused numbers from the second parent to the first child, then it does the same process to generate a second-child, but it will keep the first part of the second parent without change and copy the remaining unused numbers from the first parent to the second child *an example.*

  Parent 1 = 5 4 6 —3 1 2
  Parent 2 = 1 6 5 —4 3 2
  so, child 1= 5 4 6— 1 3 2 and child 2= 1 6 5 — 4 3 2

- multiPoint: this method is a generalization of the one-point crossover wherein alternating segments are swapped to get new offsprings *an example.*

  Parent 1 = 5 4 —6 3— 1 2
  Parent 2 = 1 6 —5 4— 3 2
  so, child 1 = 1 5 —3 6 —4 2 , and child 2= 6 3— 5 4 —1 2

- marriagekeys: The main purpose of creating this method is, as we know all of the CrossOver operators can generate N of children equals the number of parents Since the selection step gives us half of the original population Certainly the next generation will be half of the original population for that, this method receives two arrays the first array keeps all of the new children and the second array keeps (the best parent whose got it from the selection step) or apply another crossover operator to give us the second half of child, finally, it will marriage the arrays and keep them into newpopulionKey array which represents the next population.

4.2.7. Mutation Steps

under this class Mutation.java we will mutate the gene, where We can determine a specific percentage of the generation then perform the mutation process on this part of the population only, so, the Mutation constructor receives the array of the new gene which we got from CrossOver steps with a mutation ratio in population, then we can mutate 2 positions for each child, these positions are random numbers in each child *an example.*

position 1 = 5
position 2 = 1
a chromosome = 1 5 3 6 4 2
so, chromosome = 2 5 3 6 4 1

Then we can call Print method to print the array of population after mutation process.

Bibliografía

[1] Mohammad Ubaidullah Bokhari, Shadab Alam, and Faheem Syeed Masoodi. Cryptanalysis techniques for stream cipher: a survey. *International Journal of Computer Applications*, 60(9), 2012.

[2] Salvatore Mangano. Genetic algorithms. *Computer Design*, 1995.

[3] K Sindhuja and S Pramela Devi. A symmetric key encryption technique using genetic algorithm. *international journal of computer science and information technologies*, 5(1):414–416, 2014.

[4] Jason Brownbridge. Decrypting substitution ciphers with genetic algorithms. *Department of Computer Science. University of Cape Town*, page 12, 2007.

[5] repository of the project in github. `https://github.com/AbdullahTaher93/TFM`.

[6] testing of the repository of the project in github using Travis CI. `https://travis-ci.org/github/AbdullahTaher93/TFM`.

[7] Vikash Kumar Jha. Cryptanalysis of lightweight block ciphers. *Aalto University School of Science Degree Programme of Computer Science and Engineering, Master's Thesis*, 2011.

[8] Craig Smith. Basic cryptanalysis techniques. *November 17th*, 2001.

[9] Mehak Khurana and Meena Kumari. Variants of differential and linear cryptanalysis. *IACR Cryptol. ePrint Arch.*, 2015:473, 2015.

[10] William Stallings. *Cryptography and network security, 4/E*. Pearson Education India, 2006.

[11] Genetic Algorithms introduction. `https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_introduction.htm`.

[12] R Toemeh and S Arumugam. Breaking transposition cipher with genetic algorithm. *Elektronika ir Elektrotechnika*, 79(7):75–78, 2007.

[13] A Dimovski and D Gligoroski. Attacks on the transposition ciphers using optimization heuristics. *Proceedings of ICEST*, pages 1–4, 2003.

[14] Kumara Sastry, David Goldberg, and Graham Kendall. Genetic algorithms. In *Search methodologies*, pages 97–125. Springer, 2005.

[15] David E Goldberg. Using time efficiently: Genetic-evolutionary algorithms and the continuation problem. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, pages 212–219. Citeseer, 1999.

[16] Fitness function. `https://planetcalc.com/8038/`.

[17] Michael T Goodrich. Randomized shellsort: A simple data-oblivious sorting algorithm. *Journal of the ACM (JACM)*, 58(6):1–26, 2011.