Genetic Algorithm
Assignment 2 Report

Hridoy Rahman
Dept. of Computer Science
Brock University
St. Catharines, Ontario

***Abstract*—Genetic Algorithm is used in various fields of artificial intelligence and is a search technique that implements a simulation of evolution as a search heuristic to find a good solution. This paper discusses Genetic Algorithms and analyzes a Cryptography problem using genetic algorithms and various operators. This paper also compares two different algorithm configurations on a Cryptography problem and provides the result of the better configuration.**

I. Introduction

Genetic algorithm uses the idea of evolutionary computation, also based on survival of the fittest. The basis of any GAs is to implement the representation of candidate solutions, which are referred to as chromosomes; they are used to find a solution using a heuristic search. The alternate thinking could be evolution through natural selection[1]. Chromosomes are a set of genes, and genes are concrete representations of values. The following part is crucial to understanding the principle of the genetic algorithm consisting of Chromosome representation, Selection of random chromosomes, and Genetic operators such as crossovers and mutations. The algorithm simulates A string as a candidate solution, which represents the chromosomes, and uses those candidate solutions to analyze a cryptography problem. In this paper, a Genetic algorithm is used to find an approximation of a solution to an encrypted text by evolving the chromosomes with selections and genetic operators. A good solution is considered if the chromosome can decode an encrypted text, and the result is as close as the readable plain text. An evaluation function is used to determine the fitness of the chromosomes. The chromosome, in this case, is A string representation with a provided key size. Tournament selection is used to select K members (k = 2,3,4,5) from the population and find the two best chromosomes based on the evaluation function. Then, two crossover operators, such as Two-point crossover and uniform crossover, are used in two different program configurations to reach an optimal solution. The algorithm also incorporates the idea of elitism, which carries the best chromosomes from the previous generation to the next generation.

II. Background

The following section provides information on a pseudocode for the cryptanalysis problem's solution. The following program implements the Genetic algorithm used to get an approximated solution to the above-mentioned problem. The program takes a parameter of population size and max generation.

Genetic Algorithm System

```
0: procedure GA_System(popSize, maxGeneration)
0:     Genomes ← INITIAL_POPULATION(popSize)
0:     elite ← ELITISM(Genomes, eliteCount)
0:     for i ← 1 to maxGeneration do
0:         newGeneration ← new String[popSize]
0:         for j ← 0 to LENGTH(elite) do
0:             newGeneration[j] ← elite[j]
0:         for j ← eliteCount to popSize step 2 do
0:             parent1 ← TOURNAMENTSELECTION(Genomes)
0:             parent2 ← TOURNAMENTSELECTION(Genomes)
0:             if chooseCrossover equals "two point" then
0:                 child ← TWOPOINTX(parent1, parent2, Pc)
0:                 child1 ← MUTATE(child[0], Pm)
0:                 child2 ← MUTATE(child[1], Pm)
0:                 newGeneration[j] ← child1
0:                 newGeneration[j + 1] ← child2
0:             else if chooseCrossover equals "uniform" then
0:                 child ← UNIFORMX(parent1, parent2, Pc)
0:                 child1 ← MUTATE(child[0], Pm)
0:                 child2 ← MUTATE(child[1], Pm)
0:                 newGeneration[j] ← child1
0:                 newGeneration[j + 1] ← child2
0:         Genomes ← newGeneration
0:         elite ← ELITISM(Genomes, eliteCount)
=0
```

Initially, the program creates a random, unique string of genomes or chromosomes of population size as a set of candidate solutions. Then, it copies two elite chromosomes from the set of populations. Based on the evaluation function, it sorts the initial population in an ascending order and stores the first two elements in an array. Then, it runs a loop from generation 1 to max generation since generation 0 is our initial random solution to the problem. After that, It copies the elite chromosomes to the new population set. Then, it starts generating new chromosomes from elite to population size since the first two elements are occupied by elites. It selects two parents using tournament selection, which takes k members from the current population set, randomly selects two chromosomes, and returns the best chromosome. Then, based on the algorithm configuration, it chooses its crossover opera-

tors, either two-point crossover or uniform crossover. Then, it replaces the previous generation with the new generation and copies the elite of the new generation to the elite set.

III. Experimental Setup

Two data sets are used for experimenting with the implementation explained in the previous section. The data set contains a key and an encrypted text. The goal is to decode the encrypted text. Two different configurations are used to decrypt the text in the data set. One configuration uses a two-point crossover, and the other uses a uniform crossover; for two data sets, each with two configurations, five different parameter seeds were used. For data sets 1 and 2, the following parameters were used to run the code to find the approximate solution.

The parameters are the following sequentially followed, FileName, population Size, Generation Size, Probability of Crossover, Probability of Mutations and Crossover Selection Parameter.

```
GA obj1 = new GA( file, pop, maxGen, Pc,
Pm, eliteCount, chooseCrossover Operator);
```

For the following parameters, Pc = 1.0 means 100 percent of the time, crossover occurs, and 0.9 means 90 percent of the time. For the probability of mutations, Pm, it's either a 0 percent or 0.1 percent mutation rate.

For Data Set 1, Configuration: Two-Point Crossover

```
Two-Point Configuration:
GA obj1 = new GA( fname, 60, 800, 1.0,
    0.0, 2, "two point");
GA obj2 = new GA( fname, 60, 800, 1.0,
    0.1, 2, "two point");
GA obj3 = new GA( fname, 60, 800, 0.9,
    0.0, 2, "two point");
GA obj4 = new GA( fname, 60, 800, 0.9,
    0.1, 2, "two point");
GA obj5 = new GA( fname, 60, 800, 0.85,
    0.15, 2, "two point");
Uniform Configuration:
GA obj1 = new GA( fname, 60, 800, 1.0,
    0.0, 2, "uniform");
GA obj2 = new GA( fname, 60, 800, 1.0,
    0.1, 2, "uniform");
GA obj3 = new GA( fname, 60, 800, 0.9,
    0.0, 2, "uniform");
GA obj4 = new GA( fname, 60, 800, 0.9,
    0.1, 2, "uniform");
GA obj5 = new GA( fname, 60, 800, 0.95,
    0.05, 4, "uniform");
```

For Data Set 2, Configuration: Two-Point Crossover

```
Two-Point Configuration:
GA obj1 = new GA( fname, 60, 800, 1.0,
    0.0, 2, "two point");
GA obj2 = new GA( fname, 60, 800, 1.0,
    0.1, 2, "two point");
GA obj3 = new GA( fname, 60, 800, 0.9,
    0.0, 2, "two point");
GA obj4 = new GA( fname, 60, 800, 0.9,
    0.1, 2, "two point");
GA obj5 = new GA( fname, 60, 800, 0.96,
    0.14, 4, "two point");
```

```
Uniform Configuration:
GA obj1 = new GA( fname, 60, 800, 1.0,
    0.0, 2, "uniform");
GA obj2 = new GA( fname, 60, 800, 1.0,
    0.1, 2, "uniform");
GA obj3 = new GA( fname, 60, 800, 0.9,
    0.0, 2, "uniform");
GA obj4 = new GA( fname, 60, 800, 0.9,
    0.1, 2, "uniform");
GA obj5 = new GA( fname, 60, 800, 0.9,
    0.13, 4, "uniform");
```

IV. Results

A. Data Set 1 Two-Point Configuration

For each parameter that included crossovers and mutations, the program decoded the encrypted text at a reasonable amount. Whenever the mutation rate is 0, which means mutation did not occur, the fitness value halted at a local minimum and never tried to reach a lower global minimum. In this implementation, the program reaching a lower global minimum close to 0 is considered a solution to the problem, which decodes the encrypted text as close to plain readable English. For each data set, each configuration's last parameter setting, which is obj5 of each configuration, determines the best result in this report that can be achieved with this implementation of the Genetic Algorithm System.

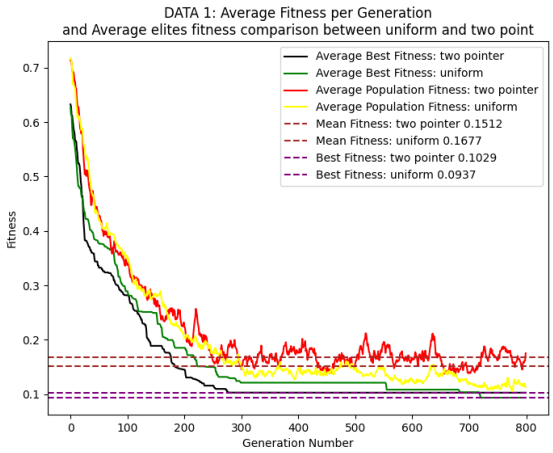
For Data Set 1, Configuration: Two-point Crossover:

The following parameter is,

```
GA obj5 = new GA( fname, 60, 800, 0.85,
0.15, 2, "two point");
```

TABLE I
Fitness Statistics with Pc = 0.85 and Pm = 0.15

Statistic	Value
Minimum Fitness	0.1050
Maximum Fitness	0.6109
Median Fitness	0.1110
Mean Fitness	0.1661
Standard Deviation	0.1102



With this parameter, it is possible to reach a solution that outputs close to an English-readable decoded text.

B. Data Set 1 Uniform Configuration

For this section as well, with a 0 percent mutation rate, it halts the best fitness at a local minimum, whereas with other parameter settings, it can reach a better result. With a mutation rate of 0.1 and a crossover rate of 1.0 or 0.9, it provides a reasonable output. The best parameter for this uniform configuration reaches 0.0937 after running the program multiple times, which outputs the decodes of the text as close to English-like.

For Data Set 1, Configuration: Uniform Crossover: The following parameter is,

```
GA obj5 = new GA(fname, 60, 0.95, 0.05, 4,
"uniform");
```

With this parameter, it is possible to reach a solution that outputs close to an English-readable decoded text.

TABLE II
FITNESS STATISTICS WITH PC = 0.95 AND PM = 0.05

Statistic	Value
Minimum Fitness	0.0937
Maximum Fitness	0.6271
Median Fitness	0.1213
Mean Fitness	0.1677
Standard Deviation	0.1019

The test for statistical significance was done with the T-Test method, Where the results suggest significant differences between the two configurations used in this implementation.

T-Statistic: -3.2729359374835068
P-Value: 0.001087090701961677
Rejects the null hypothesis. There is a significant difference between the two crossover methods.
The 'two-point' crossover method performs better.

C. Data Set 2 Two-Point Configuration

For the following data set 2, the key size is 40. Since the key size is too large, the following implementation doesn't reach close to a solution. However, it reaches a minimum value of 0.4575 with multiple experiments.

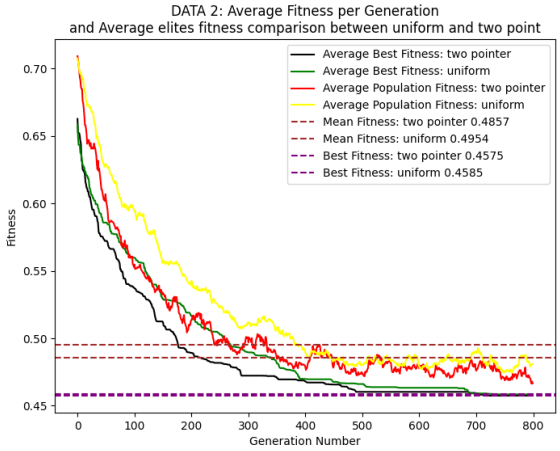
For Data Set 2, Configuration: Two-point Crossover:

The following parameter is,

```
GA obj5 = new GA( fname, 60, 0.96, 0.14,
4, "two point");
```

TABLE III
FITNESS STATISTICS WITH PC = 0.96 AND PM = 0.14

Statistic	Value
Minimum Fitness	0.4575
Maximum Fitness	0.6627
Median Fitness	0.4682
Mean Fitness	0.4857
Standard Deviation	0.0410



D. Data Set 2 Uniform Configuration

Similar results were found for this data set 2 with two-point configurations. There is only a slight difference between the performance of these two functions; none of the methods provided the desired decoded output.

For Data Set 2, Configuration: Uniform Crossover: The following parameter is,

```
GA obj5 = new GA( fname, 60, 0.9, 0.13, 4,
"uniform");
```

In this provided data set, which is data set 2, it is tough to decode the text at a reasonable amount again because the key size is high.

TABLE IV
FITNESS STATISTICS WITH PC = 0.9 AND PM = 0.13

Statistic	Value
Minimum Fitness	0.4585
Maximum Fitness	0.6586
Median Fitness	0.4699
Mean Fitness	0.4954
Standard Deviation	0.0450

For this data set, the test for statistical significance was done with the T-Test method, Where the results suggest significant differences between the two configurations used

in this implementation.

T-Statistic: -4.495322456989115
P-Value: 7.446593203224762e-06
Reject the null hypothesis. There is a significant difference between the two crossover methods.
The 'two-point' crossover method performs better.

V. Discussions/Conclusions

For every data set with two algorithm configurations, multiple experiments were performed, and statistical analysis, summary stats, and graphs were generated for all the provided parameters in the experimental setup. Then, with the best-founded data, comparisons were made between two-point crossover and uniform crossover for each of the data sets, and two final graphs were generated to show the statistical difference between the two method configurations. From the Experimental section, for all the provided parameters, data were generated. Also, the best parameters were determined over the trial and run. For data set 1, the program performed consistently and decoded the encrypted text satisfactorily. Furthermore, for data set 2, the key being too large, the program couldn't perform well. The implementation needs more optimization to decode encrypted text with higher key values. For data set 1, with both configurations, there was a satisfactory result. After doing a statistical test on both data sets, I found that the two-point crossover method was much more optimal statistically. Same with data set 2. Lastly, whenever the mutation rate was zero, the program didn't perform well, or it was stuck at the local minimum. But with ten percent mutation rates or slightly up and down, the implementation performs as expected and provides an approximate decoded text of the encrypted data. Lastly, I tried tweaking from the provided parameter, making only slight adjustments to get the possible achievable output results. However, none of them really made any difference. The parameters with a probability of crossover 0.9 and a probability of mutation rate of 0.1 provided consistent fitness value and output. But with the tweaked parameters it is possible to make a slight difference in the fitness value and lowest global was achieved with 0.937 of fitness value.

VI. References

[1] C. Darwin, "On the Origin of Species," London: John Murray, 1859.