Genetic algorithms and cryptography

Fahad Ansar
Computer Science
St. Catharine, Canada
fa16np@brocku.ca

I. Introduction

A. Concepts and topics in the report

The core concept of this report is genetic algorithm and concepts involved in genetic algorithm. Genetic algorithms have population of individual elements that keeps generating new generations using different techniques while maintaining genetic diversity, The main concepts of genetic algorithms are selection algorithm which is used for selection of parents, crossover algorithm which makes parents to exchange the characteristics and produce two new children, mutation algorithm is just for the sake of adding some diversity in children and evaluation function which evaluates the fitness of every individual in a generation. After evaluation new generation replaces old generation and goes for production of future generations.

Genetic algorithms can be used for cryptography. Cryptography is converting string or text into a code (encryption) that is unreadable without solving it using a key (decryption). Genetic algorithm can be used to find a key for decryption of an encrypted text.

B. The problem and importance of its solution

we live in a digital world. Information travels all around the world with the help of text mostly. Anyone can try to attract using different techniques like hacking and penetrating the information and see the data of anyone which can include sensitive information. This can lead to serious crimes, like violation of sensitive information. The problem can be tackled using encryption and decryption. If text is encrypted, no one can read or use it without decrypting it using the decryption key. Cryptography can work as a security lock on the sensitive data travelling through internet.If we can use genetic algorithms for generating such keys using some parameters, it would be really useful for the generation of such keys.

II. Background

A. Algorithms used in the report

1) Genetic Algorithm: It is an algorithm that is based on evolution. Genetic Algorithm simulates the evolution by improving in every generation and taking out the best out of each generation. It has four main parts, Selection algorithm, crossover algorithm, mutation algorithm and evaluation function.A typical genetic algorithm starts with generating an initial population of chromosomes that can be done using random generation or any other suitable way according to the problem. After generating the initial population, we need to select two parent chromosomes each time to perform crossover until the population size is reached.After each crossover, we need to perform mutation on both the children generated. When the population size is reached, we will evaluate every individual's fitness using fitness algorithm which will allow us to get best out of it for current generation.

After all of the main procedure of genetic algorithm, we will apply elitism on new population using previous best chromosome which is also called elite(s). We search if the best chromosome of previous generation exists in current or it has better than the best of previous generation. If new generation doesn't have chromosome with best fitness, we simply replace it with the worst fitness chromosome of new generation and if it has better than the best chromosome, the better chromosome from new population becomes our new best and we save it. If its the first generation and there is no previous best we just find the best out of it and have it saved for future generation . This whole process will get repeated for desired number of generations.

In the case of cryptography, Chromosomes would be strings of randomly generated characters from *'abcdefghijklmnopqrstuvwxyz-'*. The expected result of the genetic algorithm would be a key that can decrypt the given text. Selection algorithm would select out of randomly generated strings, Crossover algorithms swaps characters of the strings, Mutation algorithm will apply mutations on the strings (Inversion, insertion, etc), Evaluation function will evaluate the 'fitness' of the key for the given text and elitism would be performed at the end of every generation.

- Selection Algorithms: Selection algorithms in genetic algorithms are basically used for selecting parents from previous generation to produce children and form new generation.
  - **Tournament selection:** It is type of selection where we randomly select k elements from the population and select the chromosome with the best fitness out of it. The size of k can be any positive integer but recommended the size of k needs to be between 2-5. If its less than '2', selection will turn into random selection. if it is more than '5', the selection pressure increases which results to lack of diversity and lack of diversity leads to premature convergence.
- Crossover Algorithms: Crossovers is basically a phenomenon of reproduction in which we take two parents

then swaps their characteristics to produce two new children. The reason of swapping two characteristics is for children to have characteristics of both parents. Crossover is controlled by a decimal number called crossover rate. Crossover rate defines how much percentage of population will have crossover. The crossover happens using randomly generated number between '1.0' and '0.0' each time. If random number is less then the crossover rate then we select two individuals and perform crossover. If it is more then the crossover rate we don't perform crossover.

  - **One point crossover:** One point crossover is simplest crossover that can happen. In one point crossover, we take a random point in the length of parents size and swap the right or left half of the parents to produce children. This allows children to have equal characters of both parents.
  - **Uniform crossover:** Uniform crossover allows some randomness in the crossover. In this type of crossover, we generate random numbers from 0.0 to 1.0 for each alleles of parents to crossover on the basis on the number generated. After we have random numbers, we can perform crossovers on the bases of that. We simply go to each alleles and check whether it is more than 0.5. If it is more than 0.5, we swap that specific alleles else w.. e don't swap. This produces two children out of two parent depending on the random numbers.
  - **High point crossover:** I have made/designed this crossover. This crossover technique uses the values relating to chromosomes like ASCII value of string characters(In the case of cryptography). First a binary mask is needed which is nothing but binary characters in a row with its size same as the parent chromosomes. This crossover will traverse the whole string of chromosome. While traversing, At every alleles, We generate a random binary number which could be zero or one. if the binary number is '0' then don't swaps the characters between parent one and parent two . if the binary number is '1' then we check if the ASCII value of first parent character is less then ASCII value of second parent character. If ASCII value of first parent character is less then ASCII value of second parent character, we swap and if it is not, we leave it as it is. The chance of swapping is less than other chromosomes in this crossover due to double conditions for swapping.

- Mutation Algorithm: mutation algorithm is used to add some diversity and change into the children.The chromosomes are changed in a way that it doesn't match with any parent with some element of randomness. We still maintain the feasibility of the chromosome. It is done on the basis of mutation rate. Mutation rate identifies how much percent of population is gonna get mutated.
  - **Inversion:** The inversion mutation is selecting two random points in the chromosome and inverse the sub string starting from one point and ends at second point. which makes string different from its parent without turning it into a non feasible chromosomes.
- Evaluation Algorithm (Evaluation function was given): As chromosomes are strings, fitness/evaluation function checks how close is that best selected chromosome is to the actual solution of the problem which here is key string. So, the evaluation function check the occurrence rate of every single character in encrypted text and checks it with pre-saved frequency of each character of the solution.
- Elitism: it is a technique of always having the best individual till now in every coming generation by saving it and replacing it with a better one if a better one is found in future generations. Every time a new generation is generated, it is checked for the existence for the best chromosome, if new generation doesn't have it, the worst chromosome in new generation is replaced with the best chromosome we have. if it has chromosome with better fitness then the saved best chromosome is replaced with new best.

III. Experimental Setup

A. Experiment information

The base of the experiment is genetic algorithm. As using genetic algorithm, decryption can be done. First of all we need an initial population to start the generation. It would be generated randomly. To generate it randomly we need a seed for random function which would help us to keep track of randomness. So each time we give that seed to random, it will same sequence and we will get same results. After generation of initial population, we set parameters of genetic algorithm, There are only five parameters for genetic algorithm. which are mutation rate, crossover rate, encrypted text, initial population size (that would be used for generation of initial population and future populations) and key size (to constraint the size of population). We run the genetic algorithm method with all these parameters. Genetic algorithm method in each iteration of the loop, Selects two individuals and generated a random number for comparison with crossover rate. If crossover rate is more then the randomly generated number, crossover happens and produces two children for mutation, evaluation and adding into new generation. if crossover rate is less then the randomly generated number, We add two selected individuals from old generation to new generation as it is. we try different strings and parameter so that the solution is not restrict to a specific string or a specific parameter. we record average and best fitness of each generation from a single run. Then we take mean of average and best fitness from every generation of all runs with same parameters. Then we plot graphs of data to observe the differences, improvement, etc from graphs of genetic algorithms runs.

B. Algorithm parameters

Inversion mutation is used in experiment for mutating in genetic algorithm. Three types of crossover are used, One point crossover, Uniform crossover and High point crossover (which is my algorithm).

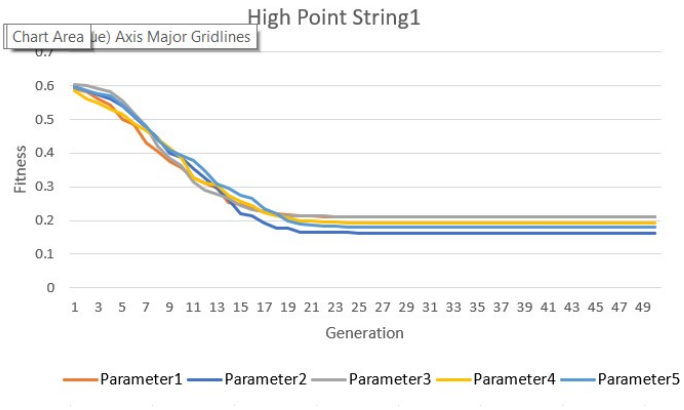

Fig. 1. String1 Highpoint crossover

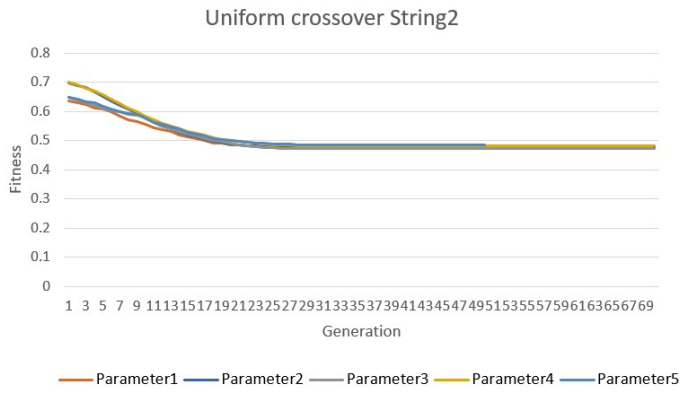

Fig. 2. String2 Uniform crossover

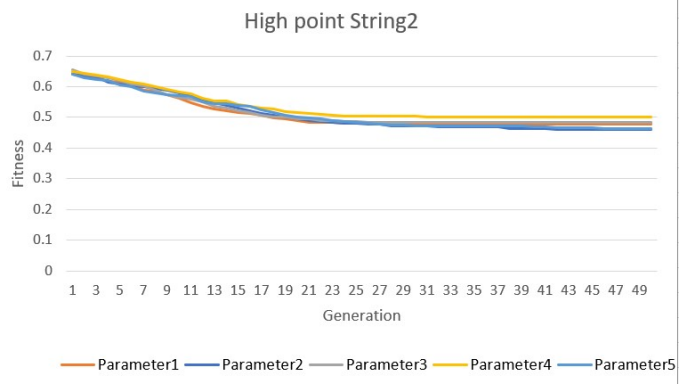

Fig. 3. String2 Highpoint crossover

IV. Results

These are findings of my experimentation in genetic algorithms.

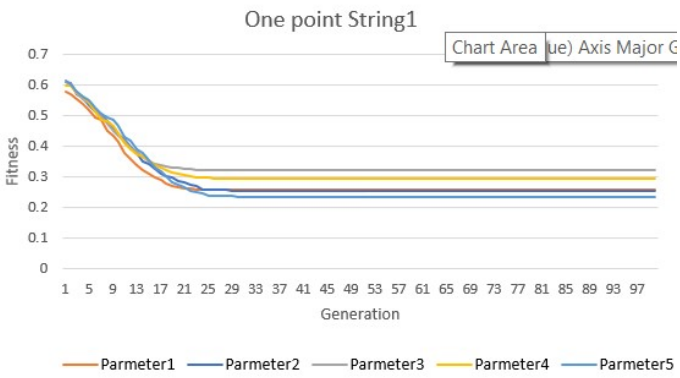

Fig. 4. String1 One point crossover

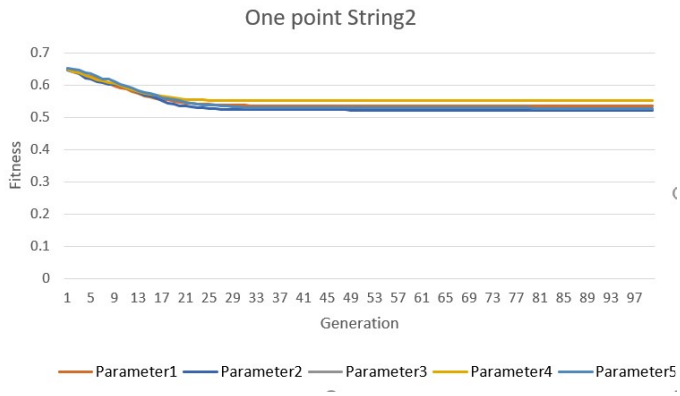

Fig. 5. String2 One point crossover

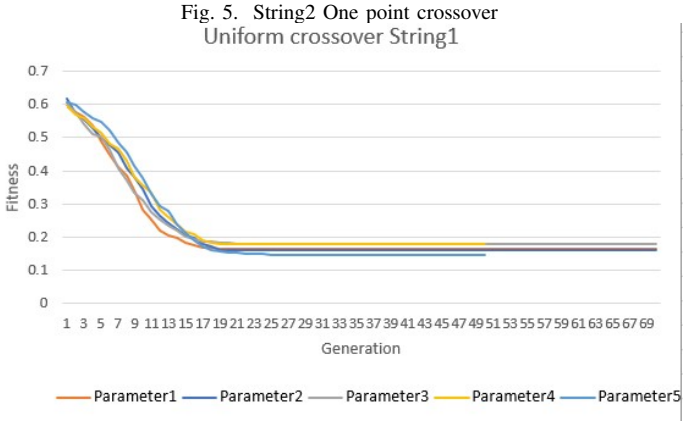

Fig. 6. String1 Uniform crossover

A. Summary of findings

- All three crossover perform different then each other. Uniform crossover perform better than both. It can be seen in fig 1 -6 that all have different graphs. If we need to apply it in decryption , we need to test differet crossover before applying and select best one.
- Genetic algorithm doesn't optimally on all parameters. It is different at 26 key size and 40 key size. As the size of the key increase it become worse. which can be seen in fig 7 and fig 8. Both graph are of different shaped. The one with less key size is more close to solution than the

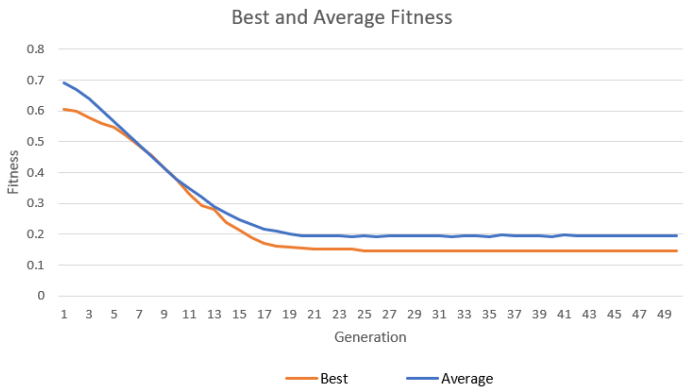

Fig. 7. String1 Solution Graph

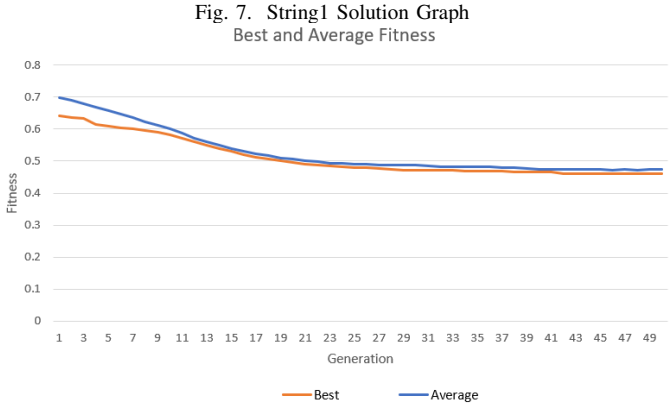

Fig. 8. String2 Solution Graph

Comparison between One point and Uniform

	Uniform	*One-point*
Mean	0.235131198	0.318311097
Variance	0.022519044	0.015004577
Observations	50	51
Pooled Variance	0.018723858	
Hypothesized Mean Difference	0	
df	99	
t Stat	-3.054429574	
P(T<=t) one-tail	0.001448549	
t Critical one-tail	1.660391156	
P(T<=t) two-tail	0.002897097	
t Critical two-tail	1.984216952	

Fig. 9. T test of Uniform and One point crossover

Comparison between High point and Uniform

	Variable 1	*Variable 2*
Mean	0.235131198	0.247191219
Variance	0.022519044	0.020091607
Observations	50	50
Pooled Variance	0.021305325	
Hypothesized Mean Difference	0	
df	98	
t Stat	-0.413117877	
P(T<=t) one-tail	0.340211387	
t Critical one-tail	1.660551217	
P(T<=t) two-tail	0.680422773	
t Critical two-tail	1.984467455	

Fig. 10. T test of Uniform and High point crossover

Summary Statistics	
Mean	0.235131198
Standard Error	0.021222179
Median	0.146778252
Mode	0.146778252
Standard Deviation	0.150063465
Sample Variance	0.022519044
Kurtosis	0.763187148
Skewness	1.515885752
Range	0.45944762
Minimum	0.146778252
Maximum	0.606225872
Sum	11.75655989
Count	50

Fig. 11. String1 Summary statistics

Summary Statistics	
Mean	0.509763594
Standard Error	0.008165046
Median	0.479579709
Mode	0.462026626
Standard Deviation	0.057735596
Sample Variance	0.003333399
Kurtosis	-0.352745648
Skewness	1.061885002
Range	0.178496496
Minimum	0.462026626
Maximum	0.640523122
Sum	25.48817968
Count	50

Fig. 12. String2 Summary statistics

other i.e String1 (26 keysize) and String2 (40 keysize).

- Doesnt give optimal solution but very close to optimal solution. As we can see in fig 7, That we are close to optimal solution but not at optimal solution.

B. Data Collected

I ran genetic algorithms five times for each parameter set, strings and crossover and then took mean of those five runs for each parameter set, strings and crossover to remove noise. Combined graphs are fig 1 - 6.

C. Final solution

There will be two final solutions. First would be from first given string and second one is from second given string. The graphs of final solution from given strings are fig 7 and fig 8.

- Both of the above the are best solution but the graphs are different from each other in context of shape and values. Form fig:1 , the final solution i found for first given string is 0.146778252. Form fig:2 , the final solution i found for second given string is 0.462026626. The summary statistics shows similar results to the graphs (fig 9 , fig 10)
- Uniform crossover performs better then other two crossovers. It is better than other two on almost all

parameters and strings. To verify that i performed T tests f Uniform crossover with other two. it shows same results (fig 11, fig12)

V. Discussions and Conclusions

A. Discussion

- According to me, genetic algorithm can be used. There are other better algorithms then genetic algorithm but it is quite close to optimal solution to a problem. The only flaw of genetic algorithm i found, a fitting parameter needs to be found for given problem to find its closed to optimal solution. There is a different kind of randomness in this algorithm.
- As per the statics (fig11,fig12), Uniform crossover performs better than high point crossover and one point crossover. It's been statically shown, in individual t-tests its is better than both.

B. Experiments and its results

Experiments were performed regarding the use of genetic algorithms for cryptography. Performed multiple experiments with different parameters, crossover and string on genetic algorithm. There are couple of things that appeared significant. First, genetic algorithm gives solution close to optimal solution. Second, with different parameters, we get really different results which means it just make multiple guesses with different parameter and evaluates those guesses generating different results each time. Third, As the key sizes increase, the performance of genetic algorithm decreases. Under light of these results, I conclude the genetic algorithms can be used for cryptography for small scale problems. But for bigger scale problem, it would take more time to reach to a parameter set that gives close to optimal solution.

VI. References

References

[1] Beatrice, Dr. Ombuki. (2019). COSC 3P71 lecture slides.