Report*

*Note: Sub-titles are not captured in Xplore and should not be used

1st Given Nitish
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

4th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

5th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

6th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—This document is a model and instructions for LaTeX. This an Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Genetic Algorithm-The genetic algorithm is a heuristic search algorithm used for solving optimization problems.It is inspired by process of genetics and natural selection.The basic idea of genetic algorithm is that it explores the solution space and improves the fitness of chromosomes with every generation.With every passing generation the algorithm improves fitness of population so that we can reach the optimal solution. Solving problem of genetic algorithm involves 6 steps which include population initialization, fitness evaluation,selection,crossover, mutation,termination.

Population-The population consist of group of chromosomes(individuals).

Chromosome-A chromosome is a possible solution to the problem

Fitness- Fitness is a parameter which is use to compare 2 solutions and find better solution.The fitter solution is selected for crossover and mutation Selection- The major purpose of selection is to select individuals from the population so that genetic operators can be used on them to find better solution.Selection of individuals is based on fitness value.There are different methods of selection like tournament selection ,roulette selection

Crossover-It is process of mating parents selected based on fitness.During this process the genetic materials of parents are exchanged which results in formation of offspring(children).The basic idea of crossover is to produce better solutions.The crossover can be of various kinds like one-point , two-point uniform and uniform -order

Mutation Random changes in genetic material is termed as mutation.It is exploration process which introduces diversity in solution and prevents solution to get struck at local optima.It can be done by various ways like inversion, displacement ,reciprocal exchange.

Termination-The steps of selection ,evaluation, crossover and mutation are repeated until termination condition is met.In the assignment the termination condition is once the loop reaches maximum generation

Cryptanalysis-The study of analyzing and breaking codes and ciphers, with the aim of understanding how cryptographic systems function and figuring out the methods to decipher them.It entails the examination and comprehension of encryption principles with the objective of identifying weaknesses and vulnerabilities that can be utilized to decode the original information without possessing the correct key.

A. Problem Statement and Why solution is important

The problem addressed in the assignment is decryption of text that has been encrypted using Vigenere Cypher.The Vigenere Cypher is a method of encrypting the text that uses a polyalphabetic substitution.The goal is to create a genetic algorithm system that can decode text encrypted using the Vigenere Cipher, advancing our knowledge of evolutionary algorithms and their application to cryptography. We can think of it from 2 different perspectives one from learning perspective and from security perspective.Implementing a GA from scratch gave me a clear picture of how GA actually works and how it can come handy in solving real-world encryption problems. Decrypting communications are a frequently used in a bunch of domains such as cryptography, law enforcement, and security.Therefore it might be helpful to know how to decode these codes for the purposes of crime investigation and evaluating the strength of encryption techniques.

II. Background

Explanation of Algorithm-

- Genetic algorithm Initializes the population with random keys. Runs a loop for the specified number of generations

Identify applicable funding agency here. If none, delete this.

and genetic algorithm runs for each generation and In each generation, performs selection, crossover, mutation, and updates the population accordingly

- Selection -Parents are selected for mating by tournament selection.Smaller the value of solution the fitter the solution.The fitness of solution is calculated by fitness() in evaluation class
- Genetic Operators-Perform crossover and mutation crossover-uniform crossover, onepoint and arithmetic mutation-reciprocal exchange Uniform crossover method-We perform crossover on 2 parents based on mask of 0s and 1s.For child1 ,if character of mask is 1 then corresponding character from first parent is selected and if character is 0 then corresponding character from second parent is selected and for child2 its vice-versa One point method-Choose a random point of crossover and for child1 choose character from index 0 to point of crossover from first parent and from point of crossover to last character from second parent and for parent 2 .Similarly for child2 choose character from index 0 to point of crossover from second parent and from point of crossover to last character from first parent Reciprcoalexchange-choose 2 random characters from string and swap them Once we get the solutions the top solutions are based to next generation.This process is called elitism.The major plus point of elitism is it helps to find best solutions
- Termination This whole of process of selection , crossover and mutation is repeated until we hit termination condition.While we are running our GA we are keeping track of the average fitness and best fitness value of both uniform and one point crossover.

A. Pseudo code-Explanation of important methods used to acheive our results

this function has all parameters that we require running our genetic Algorithm popsize-is size of population maxgen-termination condition tournamentsize-parameter for tournament selection usermutationrate-value of mutationrate(value between 0.0 and 1.0) usercrossrate-value of mutationrate(value between 0.0 and 1.0) void gaparameter(int popsize,int maxgen,int tournamentsize,double usercrossoverrate,double usermutationrate) //code **this function generates generates population of strings of the particular size** generateinitialpop(popsize,keysize) int lower = 0; final int upper = Integer.parseInt(keysize); String characters = ”abcdefghijklmnopqrstuvwxyz”; for (int i = 0; i ¡ popsize; i++)

//generate random no int lengthofrandomkeysize = randomnokeysize; generaterandomkey(); //add random key to arraylist return list

Print output for uniform crossover and one point crossover once genetic algorithm for 1 generation so outer loop will run maxgen number of times and each time inner loop of genetic algorithm runs it prints best fitness , average fitness ,best chromosome , crossover rate ,mutation rate ,number of chromosome and for both the crossover methods used for (int gen = 1; gen ¡= maxgen; gen++)

ArrayList¡String¿ uniformchilds = new ArrayList¡¿(); ArrayList¡String¿ onepointchilds = new ArrayList¡¿();

Both the arraylists above store offsprings after crossovers

// Apply genetic algorithm for (int i = 0; i ¡ popsize / 2; i++) geneticAlgorithm(parameters)

this is pseudo code for genetic algorithm.Check user crossover rate if less than randomly generated crossover rate then perform uniform and one point crossover on parents .Then condition for mutation is checked if condition is satisfied muattion is applied and if conditon for crossover not satisfied directly mutation is applied on parents geneticAlgorithm(keyArray, keysOneArray, keySize, populationSize, tournamentSize, userCrossoverRate, userMutationRate, text, uniformChilds, onePointChilds)

parent1=tournamentselection(str) parent2=tournamentselection(str) Generate a crossover mask mask = createMask(length(parent1)) Generate random crossover and mutation rates crossoverRate = randomNoBetween0And1(0.0, 1.0) mutationRate = randomNoBetween0And1(0.0, 1.0)

Check if crossover should occur if yes perform both crossovers if crossoverRate ¡= userCrossoverRate: unifCrossCh1 = uniformCrossoverChild1(parent1, parent2, mask) unifCrossCh2 = uniformCrossoverChild2(parent1, parent2, mask)

Perform one-point crossover [onePtCh1, onePtCh2] = onePointCrossover(parent1, parent2, keysOneArray)

Check if mutation should occur if mutationRate ¡= userMutationRate: Mutate the children unifCrossCh1 = mutatorFunction(unifCrossCh1) unifCrossCh2 = mutatorFunction(unifCrossCh2) onePtCh1 = mutatorFunction(onePtCh1) onePtCh2 = mutatorFunction(onePtCh2)

Add mutated children to lists addToList(uniformChilds, unifCrossCh1, unifCrossCh2) addToList(onePointChilds, onePtCh1, onePtCh2) else: Add non-mutated children to lists addToList(uniformChilds, unifCrossCh1, unifCrossCh2) addToList(onePointChilds, onePtCh1, onePtCh2) else: Check if mutation should occur if mutationRate ¡= userMutationRate: // Mutate the parents unifCrossCh1 = mutatorFunction(parent1) unifCrossCh2 = mutatorFunction(parent2) onePtCh1 = mutatorFunction(parent1) onePtCh2 = mutatorFunction(parent2)

Add mutated parents to lists for mutation addToList(uniformChilds, unifCrossCh1, unifCrossCh2) addToList(onePointChilds, onePtCh1, onePtCh2) else: // Add non-mutated parents to lists

this function generates a random key which help us to build our initial population function generateRandomKey(len, characters): sb = new StringBuilder() for i = 0 to len - 1: random = new Random() idx = random.nextInt(characters.length()) randomChar = characters.charAt(idx) sb.append(randomChar)

// Add dash after character except last if i ¡ len - 1: sb.append(”-”)

return sb.toString()

This function uses tournament selection and returns the fittest key .The fitness value is calculated using fitness function in Evaluation class function tournamentSelection(tournamentSize, keyArray, text)

// Repeat the tournament selection process 'tournamentsize' times for i = 0 to tournamentSize - 1: Randomly select a key from the key array randomIndex = getRandomIndex(keyArray) selectedKey = keyArray[randomIndex]

Evaluate the fitness of the selected key using some evaluation function evaluation = new Evaluation() fitness = evaluation.calculateFitness(selectedKey, text) Check if the fitness of the selected key is better than the current and update the value accordingly if fitness ¡ bestFitness: bestFitness = fitness fittestKey = selectedKey

return fittestKey

This method performs uniform crossover on parents and returns a offspring 1.The basic idea behind the algorithm is we generate a binary mask if character at index i in mask is 1 then corresponding character from first parent is selected and if character in mask is 0 then corresponding character from second parent is selected function unicrossoverChild1(parent1, parent2, mask): Check if parent1 is null or empty, and generate a random string if needed

Initialize an empty string for the child's genes childGenes1 = ""

Perform crossover based on the mask for i from 0 to length(mask) - 1: if mask[i] == '1': Take gene from parent1 if the corresponding bit in the mask is 1 childGenes1 = childGenes1 + parent1[i] else if mask[i] == '0': Take gene from parent2 if the corresponding bit in the mask is 0 childGenes1 = childGenes1 + parent2[i]

Return the genes of the first child return childGenes1

Convert the StringBuilder back to a string and return it

this function performs mutation . Get the characters at the randomly selected indices Swap the characters at the selected indices child1StringBuilder.setCharAt(firstIndex, secondChar) child1StringBuilder.setCharAt(secondIndex, firstChar)

Convert the StringBuilder back to a string and return it

function mutatorFn(String child): Create a StringBuilder from the input child string child1StringBuilder = new StringBuilder(child)

Get the length of the child string length = child.length() Generate two random indices within the range of the child string firstIndex = random.nextInt(length) secondIndex = random.nextInt(length) firstChar = child1StringBuilder.charAt(firstIndex) secondChar = child1StringBuilder.charAt(secondIndex) return mutatedChild

Performs one point crossover by randomly generating a random and splitting the parents from point of crossover.The result after crossover is string array of size 2 and array is returned

Initialize an array to store the two child strings String[] arr = new String[2];

Create a random number generator Random random = new Random();

Get a random crossover point within the length of the shorter parent string int pointofcrossover = random.nextInt(Math.min(parent1.length(), parent2.length()));

Create child1 by combining the first part of parent1 and the second part of parent2 String child1 = parent1.substring(0, pointofcrossover) + parent2.substring(pointofcrossover, parent2.length());

Create child2 by combining the first part of parent2 and the second part of parent1 String child2 = parent2.substring(0, pointofcrossover) + parent1.substring(pointofcrossover, parent1.length());

Store the children in the array arr[0] = child1; arr[1] = child2; return arr;(//return arr)

III. Experimental Setup

Parameters-Popsize,Maxgen,Tournamentsize,
Crossover,Mutation rate
Crossovertypes -Uniform order,onepoint
Mutation Method-Reciprocal Exchange

A. Objective

The major objective of the experiments is to measure the performance of a Genetic Algorithm (GA) in solving a cryptography-related optimization problem. The problem involves finding an optimal key to decrypt a given encrypted text. The GA is employed to evolve a population of candidate keys over several generations, with the aim of achieving better fitness values.

The experiments are for 5 different cases which involves different values of crossover rate and mutation rate for both of uniform crossover method and one point crossover

TABLE I
Experimental Data

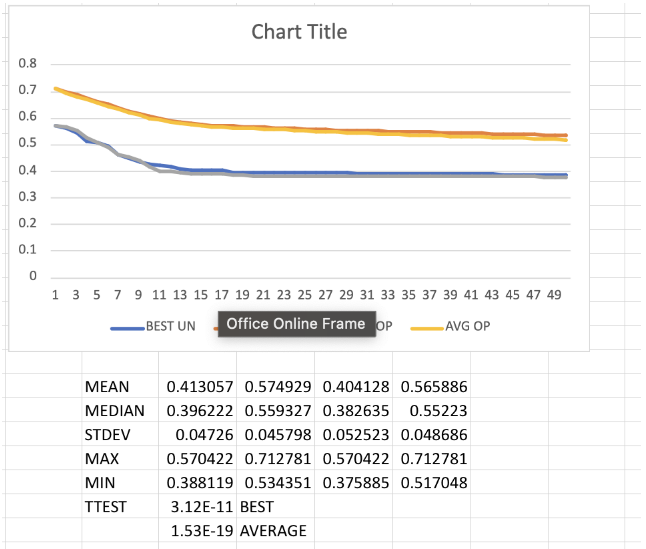
Crossover Rate	Mutation Rate
100%	0%
100%	10%
90%	0%
90%	10%
97%	10%

In each case we are trying to find best solution,best fitness and average fitness by using genetic algorithm.The Algorithm runs till we meet our termination condition of max generation.Each time algorithm runs the process of selection, crossover and mutation are done depending on conditions mentioned in code.We are keeping the track of average and best fitness of both uniform and one point crossover.Then we are comparing the best and average values of both the crossovers and plotting them on the line graph.We find maximum,minimum ,mean ,standard deviation of all of these parameters to see the variation of both the crossover methods.

B. Results

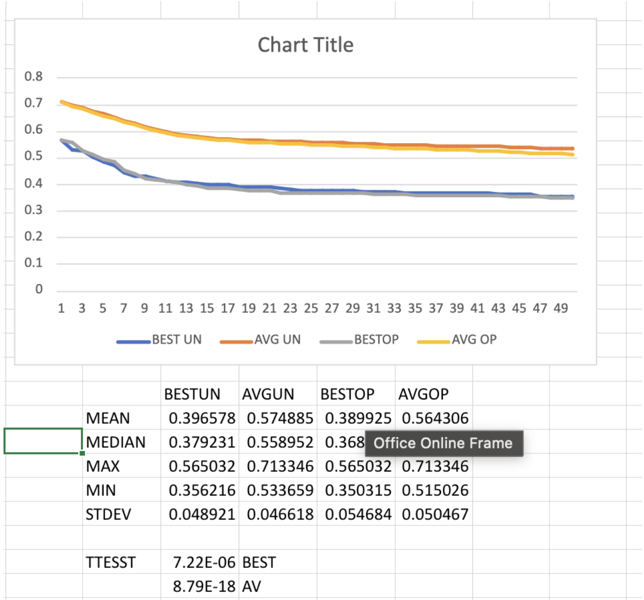
Both of the crossover performed kind of same.With successive generations the fitness value is decreasing steadily.This decrease in fitness gives us the idea to interpret that we are reaching the optimal solution.We are calculating the average fitness value and best fitness value of generations.Using the line graph in excel we see the average and best fitness of uniform and one point are giving almost same performance in most of the case.But if I have to choose one of one point is performing little better(not significant though) as the value of one point crossover in average and best case is decreasing little faster compared to uniform crossover.Since both of the crossovers are giving the same performance so both of crossover will reach the optimal solution at same point once the termination condition hits.

C. data1

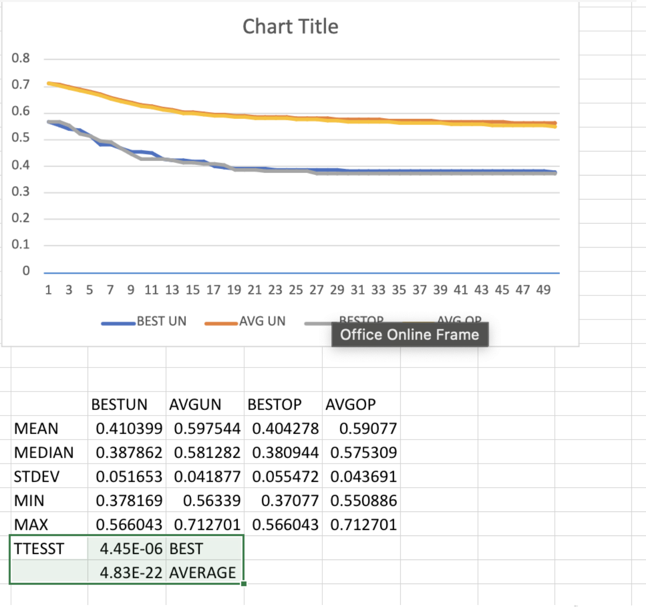


MEAN	0.413057	0.574929	0.404128	0.565886
MEDIAN	0.396222	0.559327	0.382635	0.55223
STDEV	0.04726	0.045798	0.052523	0.048686
MAX	0.570422	0.712781	0.570422	0.712781
MIN	0.388119	0.534351	0.375885	0.517048
TTEST	3.12E-11	BEST		
	1.53E-19	AVERAGE		

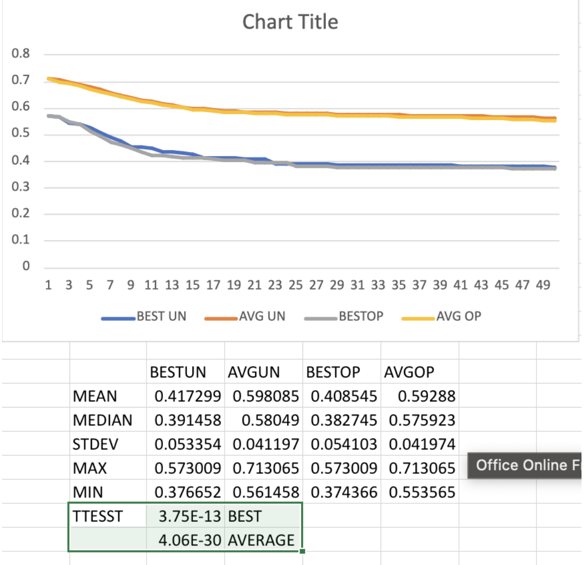
The graph depicts change in fitness with successive generation.The x axis has number of generations and y axis has fitness value.The minimum value of x axis is 1 and maximum value is 50 .



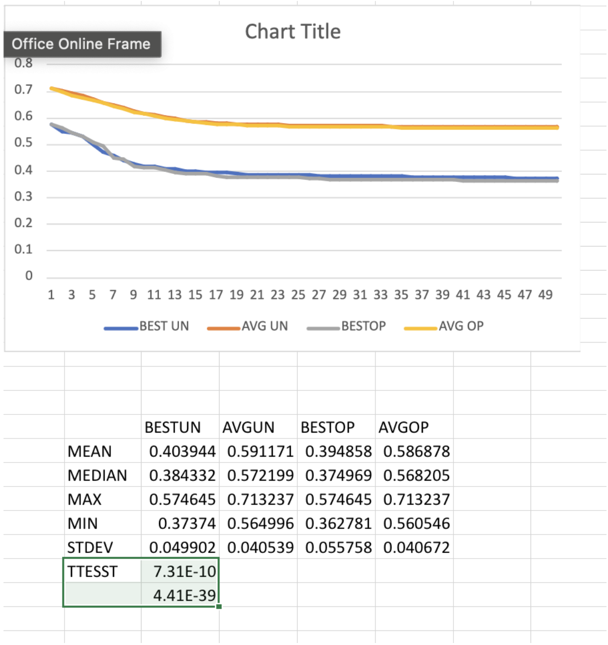
	BESTUN	AVGUN	BESTOP	AVGOP
MEAN	0.396578	0.574885	0.389925	0.564306
MEDIAN	0.379231	0.558952	0.368	
MAX	0.565032	0.713346	0.565032	0.713346
MIN	0.356216	0.533659	0.350315	0.515026
STDEV	0.048921	0.046618	0.054684	0.050467
TTESST	7.22E-06	BEST		
	8.79E-18	AV		



	BESTUN	AVGUN	BESTOP	AVGOP
MEAN	0.410399	0.597544	0.404278	0.59077
MEDIAN	0.387862	0.581282	0.380944	0.575309
STDEV	0.051653	0.041877	0.055472	0.043691
MIN	0.378169	0.56339	0.37077	0.550886
MAX	0.566043	0.712701	0.566043	0.712701
TTESST	4.45E-06	BEST		
	4.83E-22	AVERAGE		

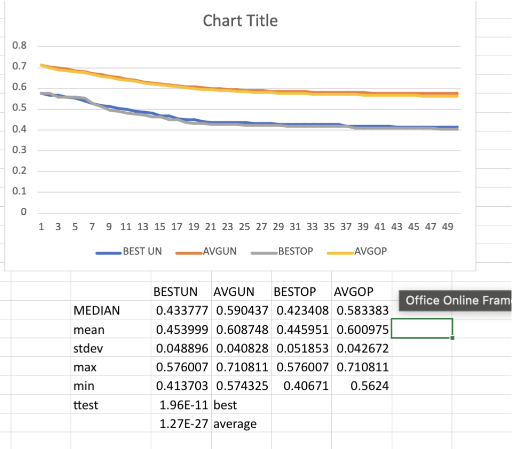


	BESTUN	AVGUN	BESTOP	AVGOP
MEAN	0.417299	0.598085	0.408545	0.59288
MEDIAN	0.391458	0.58049	0.382745	0.575923
STDEV	0.053354	0.041197	0.054103	0.041974
MAX	0.573009	0.713065	0.573009	0.713065
MIN	0.376652	0.561458	0.374366	0.553565
TTESST	3.75E-13	BEST		
	4.06E-30	AVERAGE		

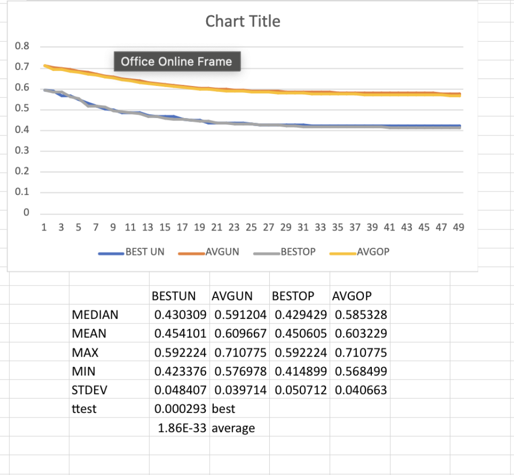


	BESTUN	AVGUN	BESTOP	AVGOP
MEAN	0.403944	0.591171	0.394858	0.586878
MEDIAN	0.384332	0.572199	0.374969	0.568205
MAX	0.574645	0.713237	0.574645	0.713237
MIN	0.37374	0.564996	0.362781	0.560546
STDEV	0.049902	0.040539	0.055758	0.040672
TTESST	7.31E-10			
	4.41E-39			

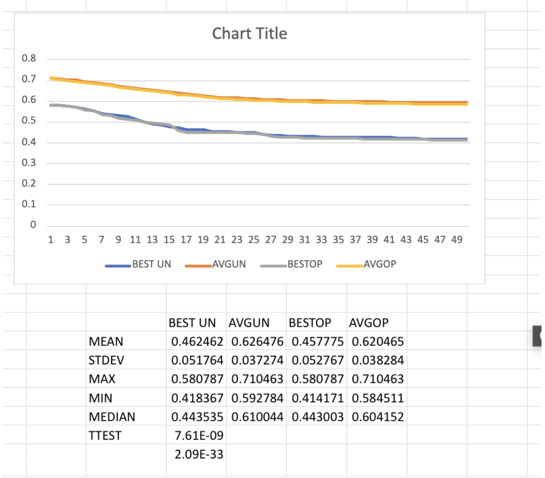
D. data2



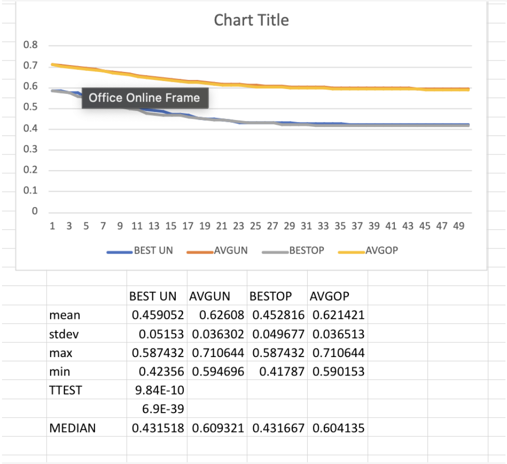
	BESTUN	AVGUN	BESTOP	AVGOP
MEDIAN	0.433777	0.590437	0.423408	0.583383
mean	0.453999	0.608748	0.445951	0.600975
stdev	0.048896	0.040828	0.051853	0.042672
max	0.576007	0.710811	0.576007	0.710811
min	0.413703	0.574325	0.40671	0.5624
ttest	1.96E-11	best		
	1.27E-27	average		



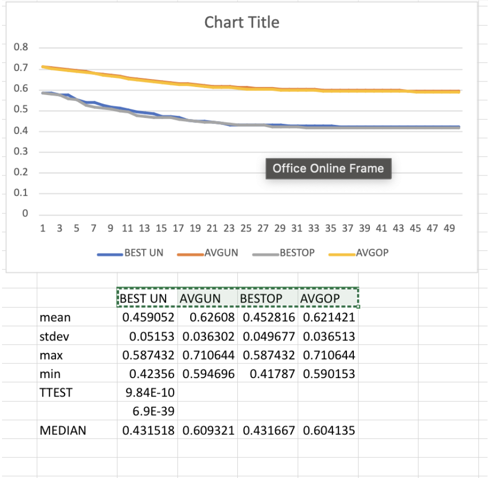
	BESTUN	AVGUN	BESTOP	AVGOP
MEDIAN	0.430309	0.591204	0.429429	0.585328
MEAN	0.454101	0.609667	0.450605	0.603229
MAX	0.592224	0.710775	0.592224	0.710775
MIN	0.423376	0.576978	0.414899	0.568499
STDEV	0.048407	0.039714	0.050712	0.040663
ttest	0.000293	best		
	1.86E-33	average		



	BEST UN	AVGUN	BESTOP	AVGOP
MEAN	0.462462	0.626476	0.457775	0.620465
STDEV	0.051764	0.037274	0.052767	0.038284
MAX	0.580787	0.710463	0.580787	0.710463
MIN	0.418367	0.592784	0.414171	0.584511
MEDIAN	0.443535	0.610044	0.443003	0.604152
TTEST	7.61E-09			
	2.09E-33			



	BEST UN	AVGUN	BESTOP	AVGOP
mean	0.459052	0.62608	0.452816	0.621421
stdev	0.05153	0.036302	0.049677	0.036513
max	0.587432	0.710644	0.587432	0.710644
min	0.42356	0.594696	0.41787	0.590153
TTEST	9.84E-10			
	6.9E-39			
MEDIAN	0.431518	0.609321	0.431667	0.604135



	BEST UN	AVGUN	BESTOP	AVGOP
mean	0.459052	0.62608	0.452816	0.621421
stdev	0.05153	0.036302	0.049677	0.036513
max	0.587432	0.710644	0.587432	0.710644
min	0.42356	0.594696	0.41787	0.590153
TTEST	9.84E-10			
	6.9E-39			
MEDIAN	0.431518	0.609321	0.431667	0.604135

E. Bonus

For bonus I have written the code of arithmetic crossover.

F. Conclusion

From the experiments on both the data1 and data2 it can be concluded both uniform and one point crossover are giving kind of same performance.But if I had to choose the better crossover , we can see one point crossover is performing little bit better.The t-test condition has very small values which is indicator that there is very low variability between best uniform crossover,onepoint crossover.Similar is the case when value of ttest was calculated for average uniform and one point crossover.

From looking at the line graph it can be concluded that the size of decryption key has a big impact on the performance of genetic algorithm and hence on the final solution. The reason is obvious because with bigger size of decrption keys the GA has to guess more values which results in bigger pool of solutions.

G. References

S. D. Immanuel and U. K. Chakraborty, ”Genetic Algorithm: An Approach on Optimization,” 2019 International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2019, pp. 701-708, doi: 10.1109/ICCES45898.2019.9002372.