T.C.

MARMARA UNIVERSITY

FACULTY of ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CSE4082 - Homework 2

GROUP MEMBERS

150115021 - Öznur AKYÜZ

150115069 - Muhammet ŞERAMET

January 7, 2020

CONTENT

1. CODE

a) GraphReader

This file reads the graph input. The readGraphFile() function is given fileName as the input parameter. After the number of nodes and edges in the first two lines of the file is taken, node weights are initialized. Then the edges in the graph are added according to the input file. Since the graph is undirected, there is no need to add the opposite after adding an edge. This is checked with the isEdgeAdded() function. A ratio is then determined by using the weight and edge number for each node so that it can be used in the repair () function.

b) Edge

We created Edge objects to indicate which nodes the edges on the graph belong to.

c) GeneticAlgorithm

i) Binary Tournament

In this function, for a given population size, we randomly select 2 individual strings from the population and determine the winner of that tournament as the individual with the lowest weight value. Weights of each individual is calculated in findWeight() function.

ii) Crossover

In this function, for a given population size, we generate a random value and compare it to the given crossover value. If the random value is greater than or equal to crossover value, we take 2 sequential individual strings (parents) from the population and randomly select a crossover point. After that, the tails of its two parents are swapped to get new individuals. Then, new individuals are added to the populationList.

iii) Mutation

In this function, for a given population size, we generate a random value and compare it to the given mutation value. If the random value is less than or equal to mutation value, we select a mutation bit by random. After that, the value of the mutation bit is flipped. Then, changed value is added to the populationList.

iv) GeneratePopulation

In this function for a given population size, we generate a random strings with the given length.

v) Repair

In this function for all strings in the population, we check its feasibility on MWVCP. If the string is feasible, we do not make any changes on string. However, if the string is not feasible, then we try to make string feasible. To make string feasible, we first locate list of indexes that store non-added vertexes (0's in the string), then from all of these located vertexes, we choose a vertex with has the lowest ratio. After that, we flip that vertex to 1 and remove from the non-added vertex list. This part is continued until the string becomes feasible.

When the string becomes feasible, then we calculate string's weight and add to the total weight of the population. Then we compare the minimum weights, if the new string's weight is less than minimum weight of the population, then minimum weight of the population is becoming new string's weight.

vi) isFeasible

To find out whether the given string is feasible on MWVCP, we check the all added-vertexes (1's in the string) and remove all the edges from the graph such that the added-vertex has. This loop is continued for all the added vertexes in the string. If we removed all the edges in the given string, then it means that string is feasible, and we return true. If there are any edges left, we say that the string is not feasible so return false.

vii) findWeight

To find out weight of the given string, we find all the added-vertexes and then calculate total sum of all their weights.

viii) runGeneticAlgorithm

It runs the genetic algorithm by calling the repair (), binaryTournament (), crossover (), and mutation () functions, respectively, by looping the numberOfGeneration value. As a result, it saves the minimum and average weight values as output.

ix) runAllCombinations

It runs the algorithm with all input parameters given in the assignment file. A total of 16 execution takes place.

d) Main

It runs the algorithm according to the command line entries. There is no error check for the inputs. The file name must be written without '.txt'. Then, 1 should be selected for homework requests and 2 should be selected to run with specific inputs.

2. EXPERIMENT

Although we continued the experiments on 3 different computers for 3 days, we were not able to get all the results. The results of the completed experiments are as follows:

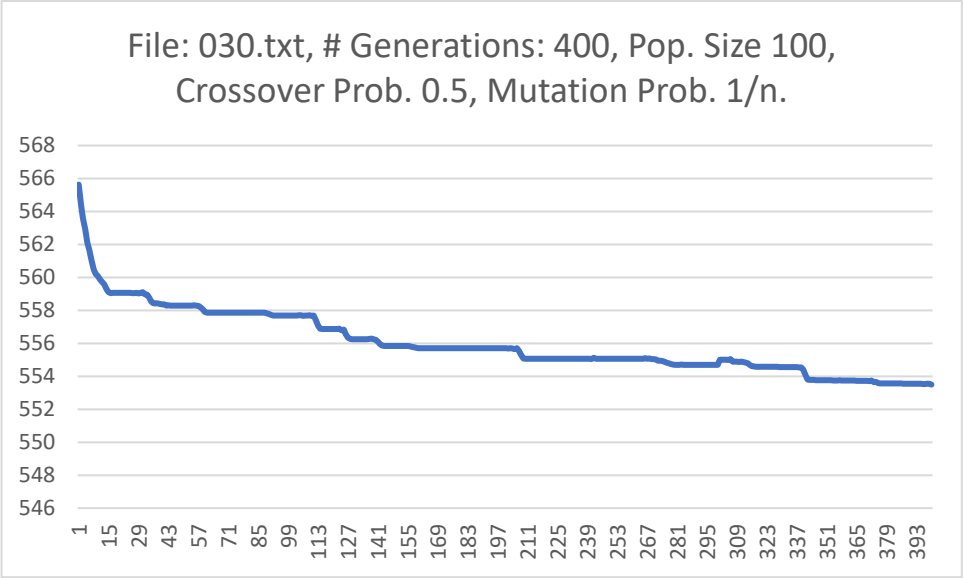
003.txt					
	Crossover Probability	**0.5**		**0.9**	
#Generations	**Pop.Size \ Mut. Pr.**	**1/n**	**0.05**	**1/n**	**0.05**
100	**100**	472,841	465,119	472,372	465,944
	200	468,621	461,980	470,731	465,433
400	**100**	470,394			
	200			467.941	

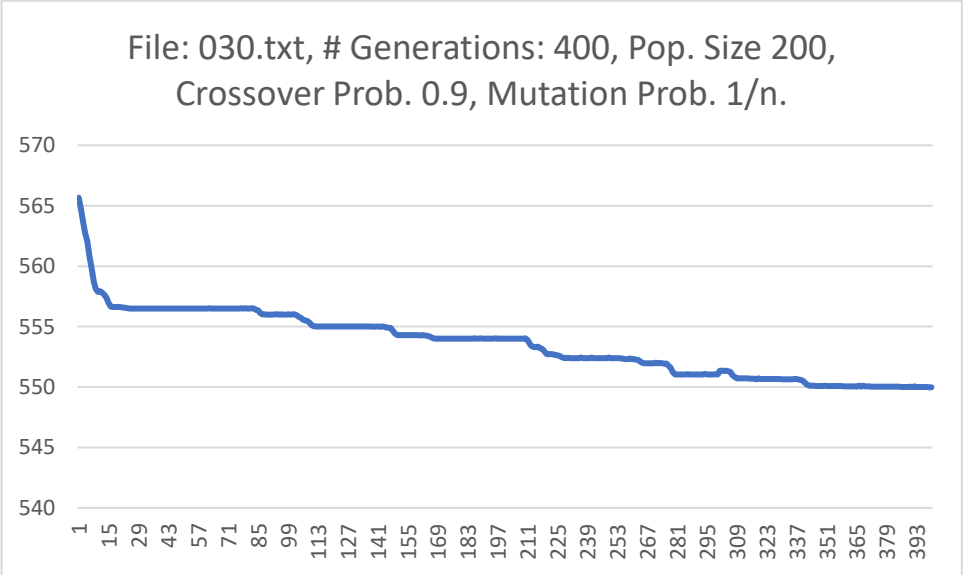
e

ex

015.txt					
	Crossover Probability	**0.5**		**0.9**	
#Generations	**Pop.Size \ Mut. Pr.**	**1/n**	**0.05**	**1/n**	**0.05**
100	**100**	473,095	465,379	472,556	466,250
	200	469,314	462,830	471,393	466,213
400	**100**	470,866			
	200			468,409	

030.txt					
	Crossover Probability	**0.5**		**0.9**	
#Generations	**Pop.Size \ Mut. Pr.**	**1/n**	**0.05**	**1/n**	**0.05**
100	**100**	558,567	550,967	558,102	551,794
	200	554,397	547,913	551,301	551,295
400	**100**	555,998			
	200			553,551	





3. DISCUSSION

a) Effects of Generation Number

Increasing the number of generations has good effects on both average weight and minimum weight. Because every new generation consists of combining information from good parents. So, we can expect better offspring from every new generation.

b) Effects of Population Size

Increasing population size mostly has good effects on the results. Because mating pool size depends on population size and increasing mating pool size provides more variety.

c) Effects of Crossover Probability

Both the average weight and the maximum weight mostly negatively affected by probability of crossover. Probability of crossover is inversely proportional to the average and the maximum weight. But we can say when we have more edges between nodes increasing crossover probability has good effects on the results. Because crossover can transform a feasible state to an infeasible state. Then our repair function may transform it into a state that has lower weight or higher weight than the original ones. It depends on a number of edges in the file.

d) Effects of Mutation Probability

Both the average weight and the minimum weight positively affected by probability of mutation generally as generation number and population size. Because mutation can introduce new information, so we find the best minimum weight with a lucky mutation. Also, there are bad results because of increasing mutation probabilities. Mutation provides variety but also corrupts some good weighted states.