Visualizing Traveling Salesman Problem Using Genetic Algorithm

Eduardo Guzman-Diaz

4942058

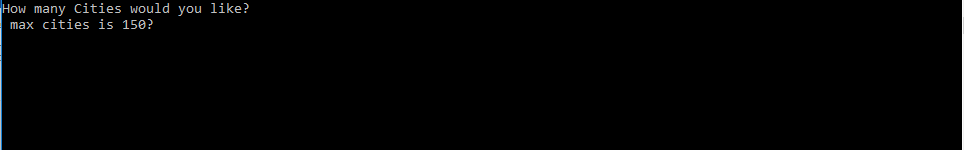
COSC 3p98

January 10, 2017

**Concept and Overview**

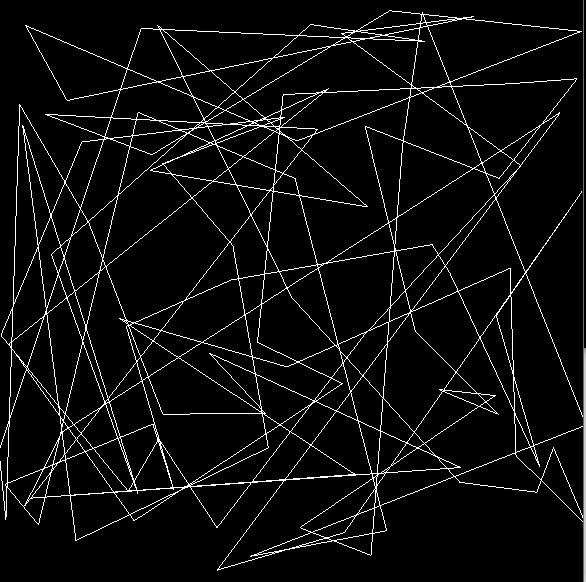
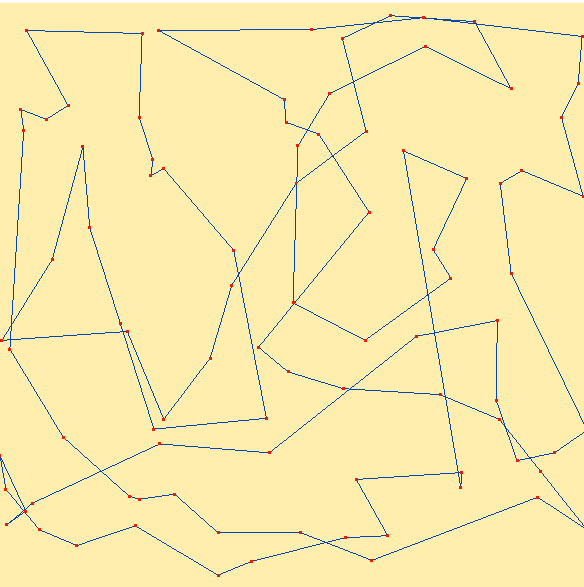
Traveling salesman problem (TSP) is -given n amount cities, what is the shortest possible path in which you can visit each city once and go back to the original starting point. There are many ways to finding local optima for this solution, I used a GA (Genetic algorithm) and showed a visualization on the process of finding a possible solution. As there are many ways in solving a TSP, A Ga provides a great visualization that progression can be seen.

**Program Overview**

Running the program is fairly simple. Once started you enter how many citied (points) you would like the GA to solve.

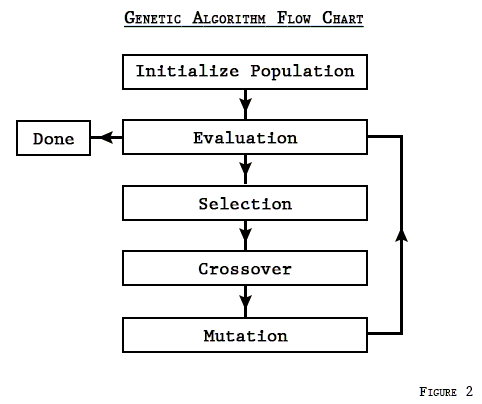
Once entered, a window will pop up showing all points generated. After, you initial the algorithm.



You can see the Initial generated path verses the final path found. We can see there is a significant difference between both paths in total path distance.

**Algorithm Overview**

A genetic algorithm is a search method that simulates evolution when finding a possible solution. Typical pseudo code look like:



Steps follow:

1)Create initial population – number of chromosomes

2)Evaluation – evaluate best solution on population

3) while max generation is not met

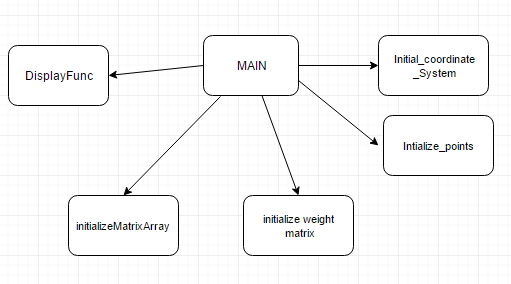
- Select new population

-implement crossover and mutation

4) evaluate best distance.

**Implementation**

Before the algorithm is initiated a few, methods are called from main class.



Initialize\_coordinate\_system(); - allocates memory for array of coordinates.

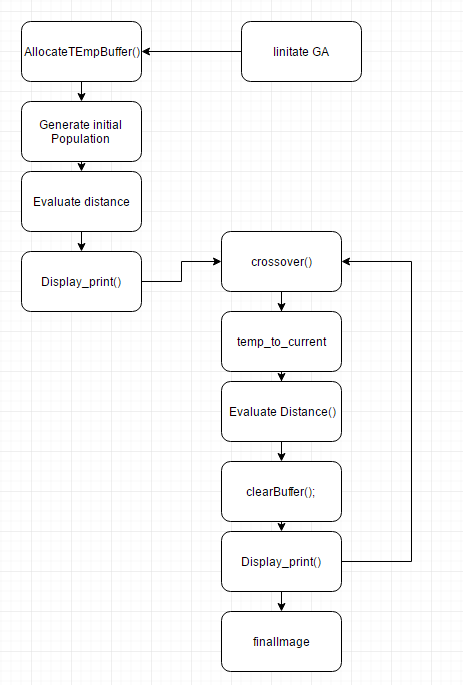
initialize\_points();- this class creates an array of coordinates based on the random generated points.

initialize\_weight\_matrix(); //allocates memory for distance array

initiateMatrixArray();- Based on the coordinates, finds the euclidean distance between all the points initialized and puts them in the matrix for easier access. (compute once and just retrieve later).

glutDisplayFunc(Display\_coordinates); displays all the points on a window.

Further after all this is calculated the GA algorithm is initiated.



allocateTempBuffer(); - allocates memory for temporary buffer.

GenerateInitialPopulation(); - Generates a random a random population.

EvaluateDistance(); - evaluates each distance of each chromosome(array).

display\_print(); - evaluates the best distance and displays it using openGL.

crossOver(); - preforms cross overs and creates and new population. Which is them put into the temp buffer.

temp\_to\_current(); - puts temp buffer to current population.

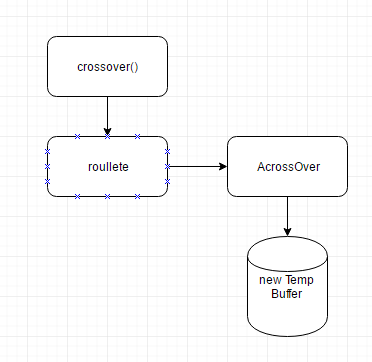
ClearBuffer(); - clears window so new route can be displayed.

finalImage(): - once final route is calculated displays it using a colour scheme for better representation using openGL.

Within the crossover method, not just the cross over happens, but so does the selection method and mutation.

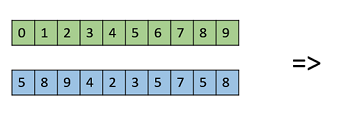
Roullete() – picks 2 chromosomes from the population using tournament selection(this will be later described in detail.

AcrossOver(): Takes 2 parents and based on thresholds, they are crossed over and mutated and put into the temp buffer of the new population. The types of mutation and cross over will later be discussed for more in depth detail.

**Pre-set Parameters**

A Genetic algorithm works on some parameters. Those being population size, max generations, mutation rate, crossover rate, nPoints.

Npoints (number of points) is the only variable the user gets to pick there is a limit for cities(max 150) because the higher the points, the more generations would be needed and we have pre set the max amount of generations. Population size was set as 251 different chromosomes. Max generations were set 1500. With 1500 generations, the user can see the progress of high number of cities clearer allowing reasonable end result from higher amount of cities. Mutation rate was set at 15%. Cross over rate was set at 90%.

**Methods** There are wide variety of methods for selection methods, crossover types and mutation methods. To mate 2 chromosomes, I used a random selection method for finding 2 parents. Throwing a random variable and moding it by number of cities. Once the 2 parents are found, the cross over method used was a uniform cross over. We create a random bit mask and flip the two parent’s chromosomes that contain a 1.



Cross over happens only when a certain threshold is met (discussed earlier). For mutation method, we selected 2 random cities within a chromosome and swapped them. Again, only happened if a certain threshold was met.

**Program Details**

This program was written in C++ and can be ran in Microsoft Visual Studio.

**Work Cited**

Armstrong, Joseph. "The Genetic Algorithm - Explained". *Techeffigytutorials.blogspot.ca*. N.p., 2017. Web. 11 Jan. 2017.

"Genetic Algorithms Crossover". *www.tutorialspoint.com*. N.p., 2017. Web. 11 Jan. 2017.