Genetic Algorithm
Lomus Hona
[Computer Science and Engineering]
Speed School of Engineering
University of Louisville
l0hona01@louisville.edu

Code:
I have commented out function call to selection and crossover algorithms (will have to uncomment based on which selection and crossover algorithms to run)

```
190
191         #--------------UnComent For RouletteWheel Selection----------------------
192             #parent1 = individuals[rouletteWheel(roulette_pie)].path
193             #parent2 = individuals[rouletteWheel(roulette_pie)].path
194         #----------------------Uncomment for TOurnament Selection----------------
195             parents = tournamentSelection(individuals, TOURNAMENT_SIZE)
196             parent1 = parents[0]
197             parent2 = parents[1]
198         #---------------------UnComment for Ordered Crossover---------------------
199             child = orderedCrossover(parent1, parent2) #offspring path of parent
200         #-------------------Uncomment for cycle crossover-----------------------
201             #child = cycleCrossover(parent1, parent2)
```

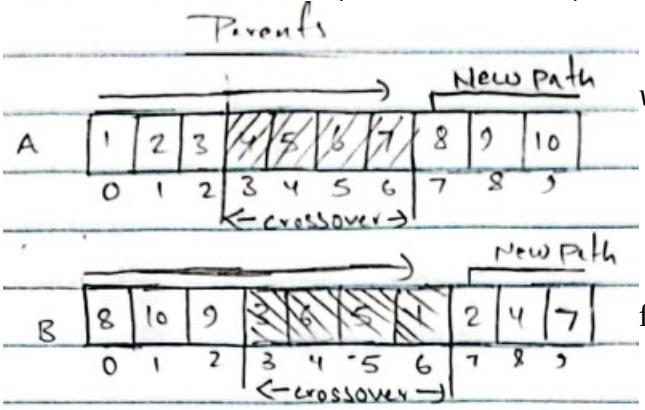
1. What did you do in this project and why?

For this project I implemented two crossover methods, two parent selection method, and one mutation method for crossover. The crossover methods I implemented are ordered crossover and cycle crossover, along with roulette wheel selection and tournament selection to select the parents, and vary basic mutation method where path to and from two randomly selected cities were swapped.

In order to run the algorithms mentioned above, I first got all the nodes from tsp file and uploaded them in an array as an instances of a class node with propertied Id (representing the node), x-coordinate, and y-coordinate. Utilizing the array of class nodes I generated a 1st generation of random paths and placed it into array named individuals, with each individual being a instance of class individual with properties path, path_distance, and path_fitness. And utilizing the three properties of individual class to be utilized by both parent selection and crossover algorithm, with 2% mutation rate. With everything mentioned above, I was able to generate all the children for each generation and picked stored best of each generation in an array to be displayed.

2. Describe the algorithm you are using for this project, Define your stopping criteria?

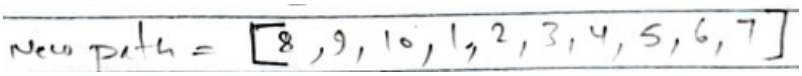
- *Ordered Crossover (returns 2 children)*



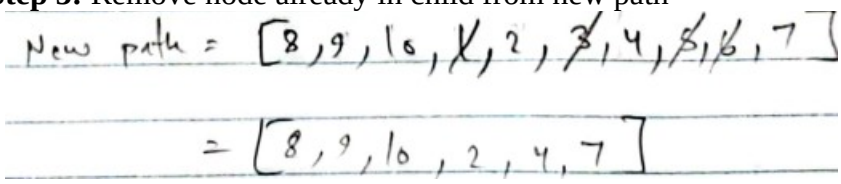
** For a given parents to the left, creating a child with more chromosome from parent A*

Step 1: Take crossover chromosome from parent B

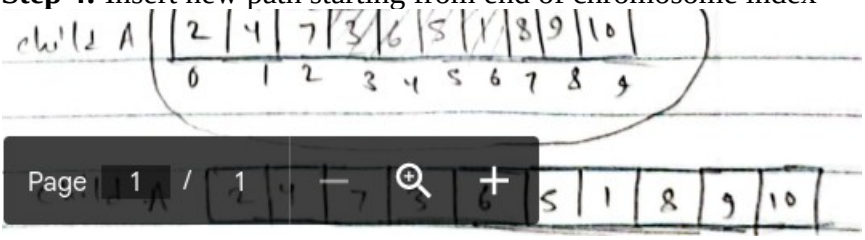
Step 2: For parent A, List new path order starting from end of chromosome index



Step 3: Remove node already in child from new path



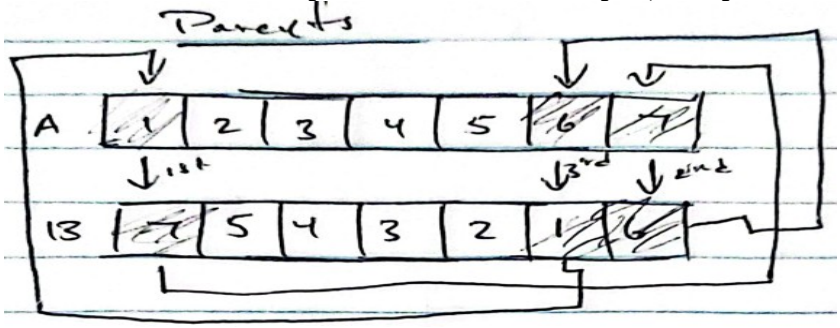
Step 4: Insert new path starting from end of chromosome index



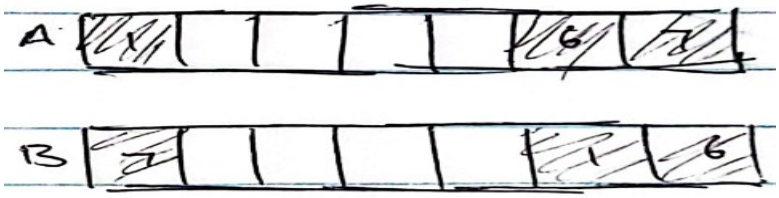
Step 5: Produce child2 in same manner but taking most of the chromosome from parent B and crossover from parent A

- *Cyclic Crossover (return 2 children)*

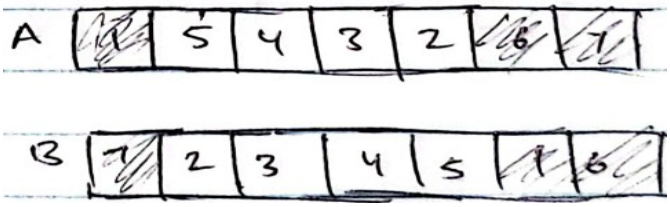
Starting from first node of parent A mapping parent B, then finding mapped value in parent B in parent A to map parent B again until value that we started from is being mapped by parent B is formed give chromosome from both parent that are to be kept. (example shown below)



chromosome to be kept from both parents for respective child:



from there rest of the non selected chromosomes are swapped between parents producing a child.



- *Roulette Wheel Selection (return 1 parent)*

For roulette wheel selection, the fitness for each path were calculated and converted to occupy part of a wheel with value ranging from 0 to 1. it was then stored in an array (acts as a roulette wheel). To spin the wheel, I used pythons random number generator to generate values between 0 and 1 that will give values upto 6 decimal points, and based on the value it gave, a parent was selected if random value generated was between array[x] and array[x+1].

- *Tournament Selection (return 2 parents)*

Chooses a parent with best fitness from a pool of 10 randomly selected parent. It does it twice to give parent 1 and parent 2 at the same time.

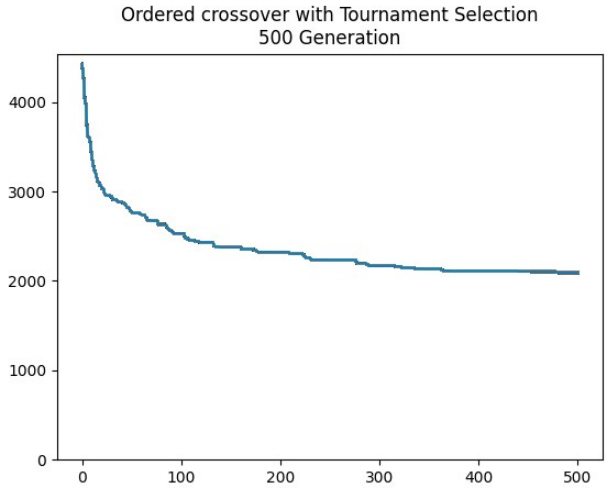
- *Mutation*

The mutation rate for the program was fixed rate of 2%. And the algorithm itself was just randomly selecting 2 chromosome and swapping them.

3. Identify your best solution for the solved problem, Analyze the effectiveness of the two chosen variations in GA, Briefly discuss how long a typical run took for each of the datasets, Graphically present your improvement curves

→ All combination of selection and crossover algorithm ran in less than a minute for starting population of 100 (each generation get 100 individuals), and 500 generation.

- Ordered crossover with tournament selection (video provided separately)

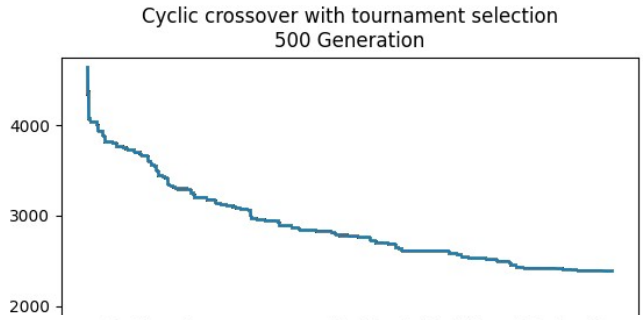


Best initial distance vs best final gen distance

```
4551.759896051262 1950.4820662056675
```

→ This combination of ordered crossover and tournament selection produced the best result thus far compared to others, with sharp decline in distance until around x = 100 and gradually slowing the change in distance down.

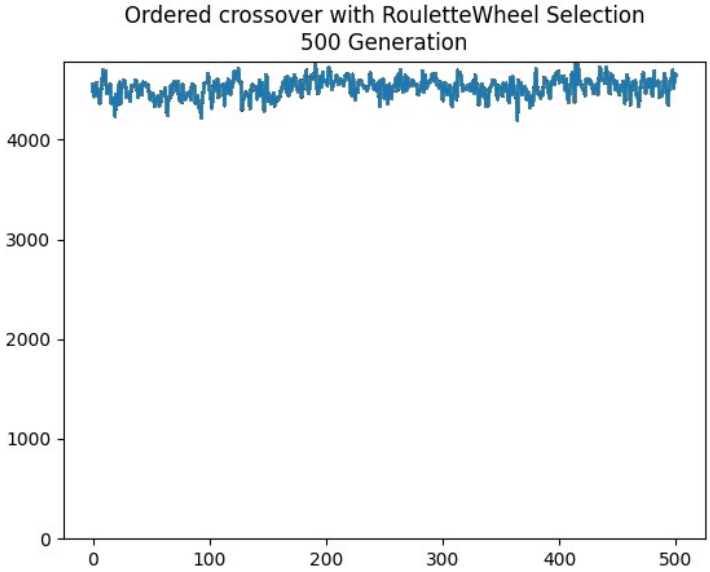
- Cyclic crossover with tournament selection (video provided separately)



Best initial distance vs best final gen distance

```
4508.823869396894 2333.080877880838
```

→ The graph is quite similar to ordered crossover, if ordered crossovers graph was shifted left to start at around 100 generation. It is overall less effective compared to ordered crossover.



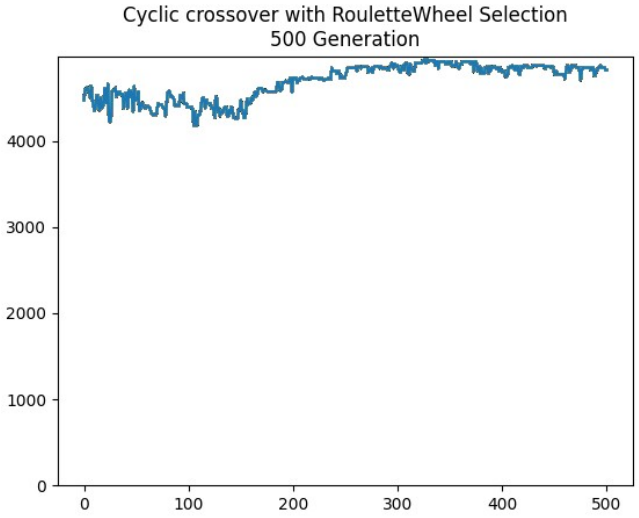
- Ordered crossover with roulette wheel selection (video provided separately)

Best initial distance vs best final gen distance

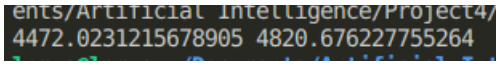
```
4549.917912863355 4638.231785767543
```

I don’t know if I did something wrong or if basic roulette wheel selection doesn’t work very well with tsp, especially if it has too many population, or perhaps it could get biased with less variance. I debugged through the algorithm couple of time but didn’t find any issue. Roulette wheel crossing simply didn’t work for me for tsp

- Cyclic crossover with roulette wheel selection (video provided separately)



Best initial distance vs best final gen distance



Same as ordered crossover with roulette wheel selection.

4. Discussion:

The final result for me would be combination of tournament selection and ordered crossover as roulette wheel doesn’t really seem to work, while tournament selection lowered the distance nearly to half in 500 generation. As for crossover method, ordered crossover beats cyclic crossover in both roulette wheel selection and tournament selection.

5. Critical thinking

The biggest problem implementing this project was with roulette wheel selection which seemed very simple but it doesn’t seem to work, or maybe it is working correctly though I don’t think so because it is increasing distance or doesn’t lower the distance by much at all. I would probably do more research on the algorithms I am implementing or try to create my own version (most likely wont work) if I have enough time. Despite poor results in some cases, I think the best I could take away from this project is the knowledge that even if there is no black and white solution, there exists or algorithms can be created in such a way to give us possible answer or direct us to good answers via reward system.

References:

https://www.youtube.com/watch?v=7hDZyH2E4Yw&t=7s
https://www.youtube.com/watch?v=e7ozr4sARyI
https://www.obitko.com/tutorials/genetic-algorithms/selection.php
https://www.geeksforgeeks.org/tournament-selection-ga/
https://www.youtube.com/watch?v=lOajq3X4WJQ&t=320s