Assignment 4:
Optimization with Genetic Algorithms
Neuronal and evolutionary computing
2023 - 2024

Pawel Puzdrowski

2024-01-XX

Contents

0.1 Description of the chromosome, and adaptations done to the algorithm

0.1.1 Translation of the problem to chromosome

Firstly the dataset is imported with help of tsplib95 library. A simple tsplib95.load() function is used to import the dataset and get proper structure of the problem so it can be handled by the genetic algorithm. The genetic algorithm has several stages and this stages are inspired from geeksforgeeks source [2]

- Initialization

- Fitness score calculation

- Elitism

- Selection

- Crossover

- Mutation

- Population update

At the Initialization stage the original population is created with create_initial_population() function. The cities are extracted from the dataset and tours are created randomly.

Fitness score calculation is done with fitness() function where following fitness score calculation is done:

$$\frac{1}{calculate_distance()}$$

Function calculate_distance() is basically calculating the total distance of a tour. Then if the distance is high the fitness score will get low and if the distance is low the fitness score will get high. Every tour is assigned one fitness score.

The part where elitism is done, it's taken from this source [3]. Elitism is done with elitism() function. The best tour from the population is chosen and added to the new population. This way the best tour is always saved in the new population and the evolution is faster.

Selection is choosing the parents for the mating part. Depending on the method the parents are choosen differently. Crossover section is creating the children from the chosen parents and depedning on the method the childrens inheritance of parents genetics is done. Mutation part has the responsibility to perform mutations on the children and as well mutation has different methods to perform mutations so the mutations can vary.

To finish off the algorithm, all the parents creates a new child and at the end a new population has arrised and its selected to do the same mating process again depending on how many generations are specified.

The different selection methods were taken from this source [3] and the different selection methods are listed below

0.1.2   Methods for selection/crossover/mutation

Selection methods: [3]

- Fitness proportionate selection - parents are choosen randomly, but the parents with higher fitness score has higher chance of being selected

- Tournament selection - 4 parents are choosen randomly from the set and the parent with highest fitness score and the second highest score is chosen to mate.

Crossover methods: [4]

- One point crossover - a randomly point is chosen on the tour and the tour is splitted in two parts. Visualization of one point crossover is shown in figure 1

- Uniform crossover - each gene (city) is treated seperately instead of divided the parent in to one or more parts. Basically a flip-coin is done on every city if its included or not. Visualization of uniform crossover is shown in figure 2

Mutation methods: [5]

- One bit flip mutation - only one bit is mutated which in this case means that one city is swapped with another random city from the same tour. Visualization of one bit mutation is shown in figure 3

- Scramble mutation - a random set of cities is chosen and swapped with another random set of cities. Visualization of scramble mutation is shown in figure 4
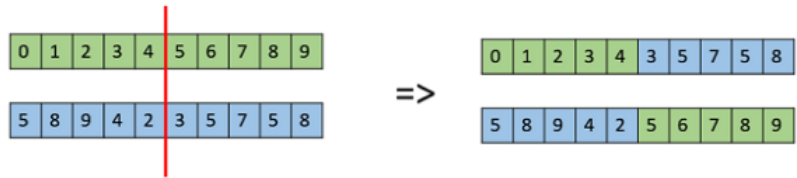
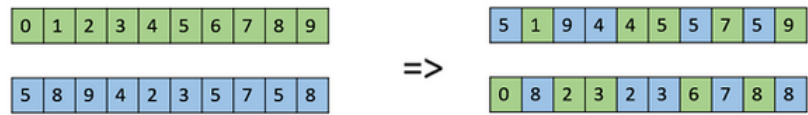| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

| 5 | 8 | 9 | 4 | 2 | 3 | 5 | 7 | 5 | 8 |

=>

| 0 | 1 | 2 | 3 | 4 | 3 | 5 | 7 | 5 | 8 |

| 5 | 8 | 9 | 4 | 2 | 5 | 6 | 7 | 8 | 9 |

Figure 1: One point cross

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

| 5 | 8 | 9 | 4 | 2 | 3 | 5 | 7 | 5 | 8 |

=>

| 5 | 1 | 9 | 4 | 4 | 5 | 5 | 7 | 5 | 9 |

| 0 | 8 | 2 | 3 | 2 | 3 | 6 | 7 | 8 | 8 |

Figure 2: Uniform crossover

| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |

=>

| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
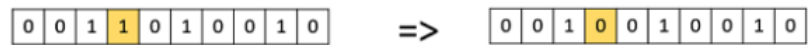
Figure 3: One bit mutation

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

=>

| 0 | 1 | 3 | 6 | 4 | 2 | 5 | 7 | 8 | 9 |
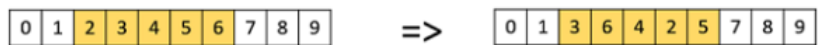
Figure 4: Scramble mutation

Population size	Generations	Mutation	Selection	Cross	Mutation	Min Dist

Table 1: Result dataset smallDataset.tsp

0.1.3   How the population size is chosen

After doing coople of test with population size, the best and efficient results are obtained when the population size of the size of the dataset or little bit higher. Double, tripple or higher population size doesn't result in better path.

0.1.4   How stationary state is identified

One of the downloaded datasets have specified on the website the lowest distance possible. Dataset att48.tsp has lowest distance of 33523, according to the website [1]. To identify stationary state for the other datasets various tests were performed on the parameters: population size, number of generations, and mutation rate, and then checking the plot of evolution of the minimum total traveling distance.

0.2   2.

0.2.1   Description of the datasets

0.2.2   The results obtained with at least 6 different combinations of parameters described in the previous section

Bibliography

[1] Data for the traveling salesperson problem. Link to source. Accessed 2024-01-05.

[2] Geeksforgeeks. Traveling salesman problem using genetic algorithm. Link to source, 2023. Accessed 2024-01-05.

[3] Eric Stoltz. Evolution of a salesman: A complete genetic algorithm tutorial for python. Link to source, 2018. Accessed 2024-01-05.

[4] Tutorialspoint. Genetic algorithms - crossover. Link to source. Accessed 2024-01-08.

[5] Tutorialspoint. Genetic algorithms - mutation. Link to source. Accessed 2024-01-08.