Time Series Prediction with Genetic Algorithms

Yi Liu - 1605486

In this Assignment, I was using python as programming language to implement the genetic algorithm for solving time series prediction. I have also tested my programme with different parameters to determine the impact of parameters on the solution quality. For each parameter setting, I did 100 independent runs. I will describe my pseudo-code first, and then show you test data and plots.

Ex.3: Pseudo-Code

1. Main Function:

```
function genetic algorithm(size,vector size,data size,data,time budget):
    time = time()
    population set,fitness set =
        population(size,TREE DEPTH,dimention,data size,data)
    generation = NUM GENERATION
    for i in range(0,generation)
        if time()-time > time budget
            break
        else:
            parent1 = random select(population set)
            parent2 = random select(population set)
            offspring = crossover(mutation(parent1),mutation(parent2))
            offspring fitness = fitness(offspring)
            population set += offspring
            fitness set += offspring fitness
        population set = ranking(population set)
        fitness set = ranking(fitness set)
       fittest = population set[0]
    return fittest
```

2. Population:

```
function population(size,max depth,vector size,data size,data):
      population set = []
      fitness set = []
      for i in range(0,size):
              individual = initialise tree(max depth)
              fitness = fitness(individual,vector size,data size,data)
              population set += [individual]
              fitness set += [fitness]
      return population_set,fitness_set
```

3. Crossover:

```
function crossover(parent1,parent2):
      node1 = random select a node(parent1)
      node2 = random select a node(parent2)
      subtree1 = get_subtree(node1)
```

```
5       subtree2 = get subtree(node2)
6       parent1[subtree1] = replace subtree2
7       offspring = parent1
8       return offspring
```

3. Mutation:

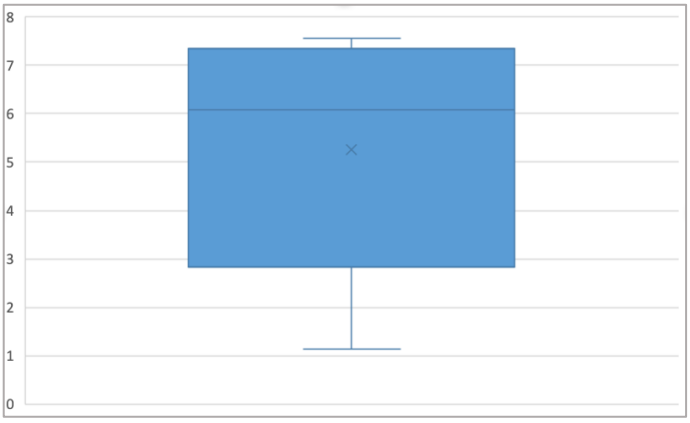
```
1 function mutation(parent)
2       if (RandomNumber < MUTATION RATE):
3               node = random select a node(parent)
4               subtree = get subtree(node)
5               mutated subtree = initialise tree(SUBTREE DEPTH)
6               parent[subtree] = replace mutated subtree
7               offspring = parent
8       return
```

Ex4. Results

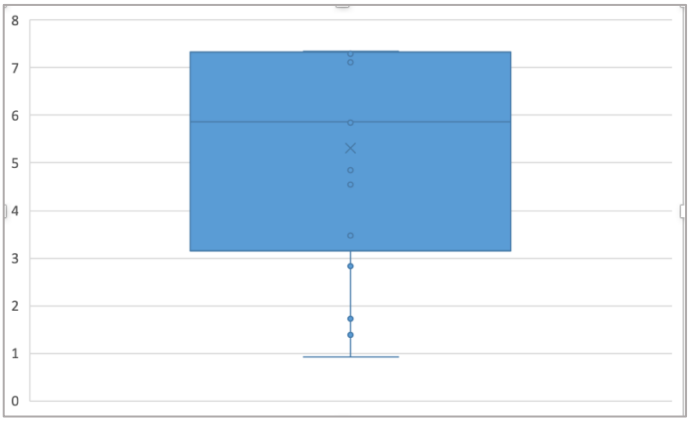
For my genetic algorithm, I tried to test the impact of different parameters on the solution quality. I did 100 runs for each parameter setting, and plotted the graph as the following.

Parameter: maximum tree depth of initialised population.

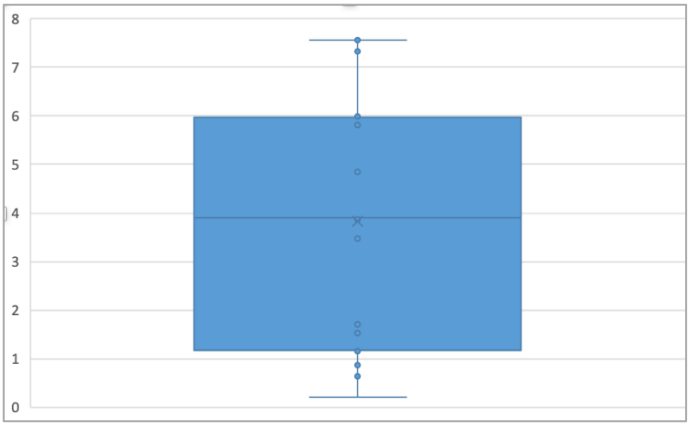
Tree Depth = 2:



Tree Depth = 3:

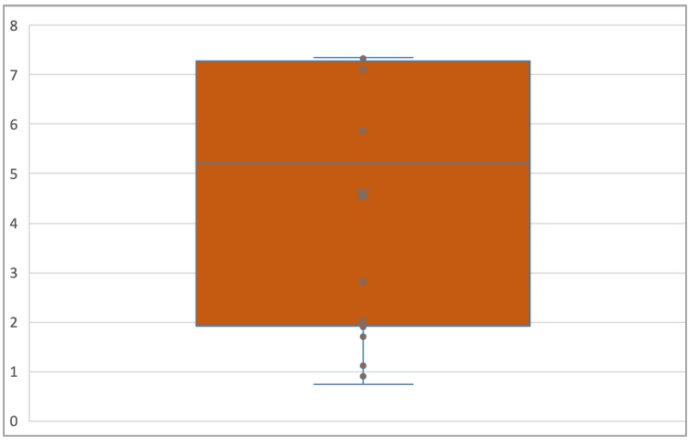


Tree Depth = 4:

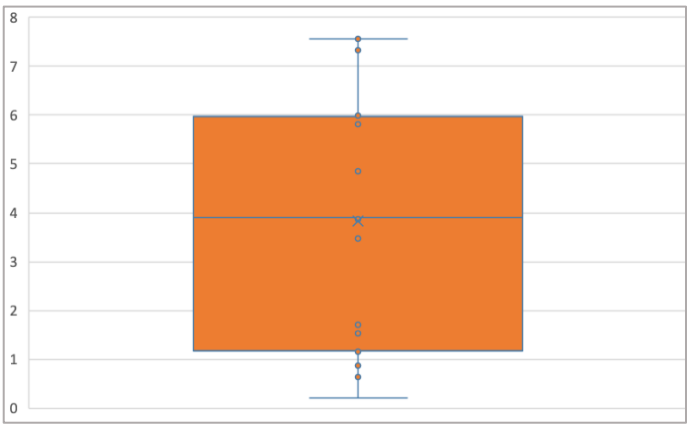


Parameter: the probability that mutate a specific individual.

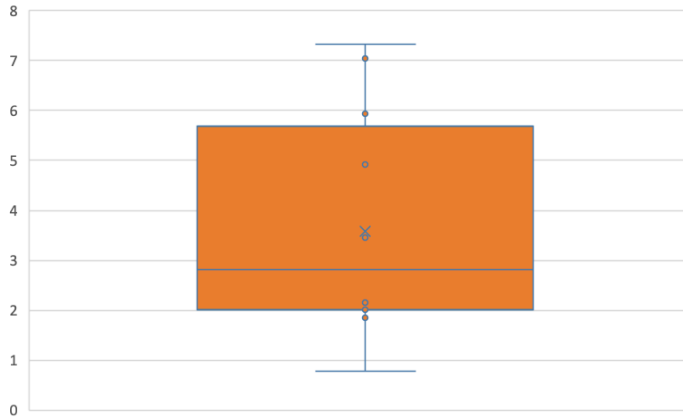
Mutation Rate: 0.25



Mutation Rate: 0.5

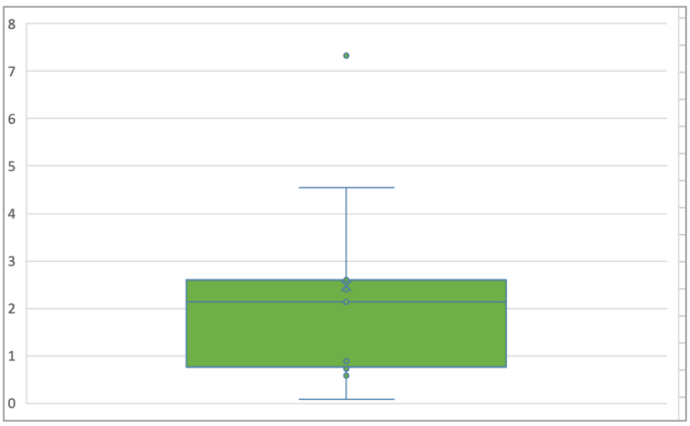


Mutation Rate: 0.75

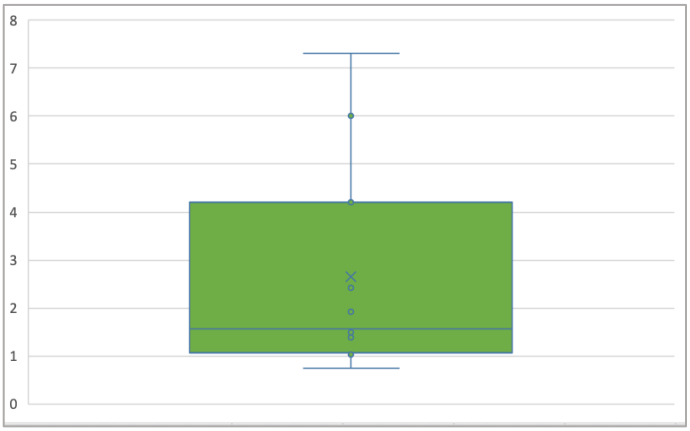


Parameter: the size of population.

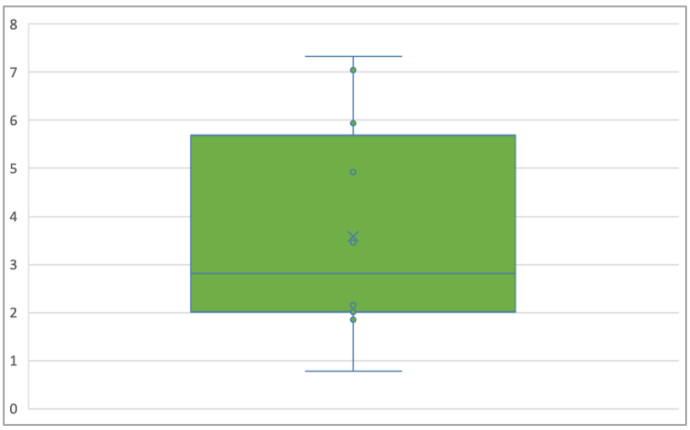
Population Size: 50



Population Size: 75



Population Size: 100



How to Run

If you want to test the programme without docker, you can use my provided data which is "data.txt" and uncomment the print function on line 551 and 540. If you want to use your own data, please change the information on line 547.