Lab 4: Time Series Prediction with GP

Student Name: Binjie Zhang Student ID: 1870958

Exercise 4

Algorithm 1: Genetic Programming

```
Input: population size, dimension, data size, data name, time budget
Output: expression
Set: crossover probability, mutation probability
Start counting time
Pop = initialization (population size)
For I in the range of population size do:
  Calculate the fitness
End for
While time<time budget:
  Parent = pop [0:population size]
  Offspring = parent
  For I in the range of population size/2:
    If random < crossover probability:
      Do crossover
      Return new offspring
  End for
  For I in range of population size:
    If random < mutation probability
      Do mutation
      Return new offspring
  End for
  For I in range of population size:
    Calculate the fitness of new offspring
  End for
  New population = pop + offspring
  Sort the new population
End while
Return best expression
```

Algorithm 2: Spanning tree

```
Input: depth, max depth
Output: tree list
Depth = depth + 1
If depth = 0:
  Tree list = []
  New node = random operator
  Add new node to tree list
  For I in the range of new node
    Spanning tree in tree list
  End for
```

```
Else if depth > max depth :
  Return random number in 0-10
Else:
  Tree list = []
  New node = random operator
  Add new node to tree list
  For I in the range of new node
    Spanning tree in tree list
  End for
Return tree list
```

Algorithm 3: Replace

```
Input: tree, position, new branch, index
Output: tree index
If position = index:
  Tree = new branch
Else:
  For I in the range of the length of tree:
    If type of tree[i] = list:
      Index = index + 1
    Replace the tree
  End for
Return tree, index
```

Algorithm 4: Find subtree

```
Input: tree, position, index, sub tree
Output: tree, sub tree, index
If position = index:
  Sub tree = tree
Else:
  For I in the range the length of tree:
    If type of tree[i] = list:
      Index = index + 1
    Find sub tree
  End for
Return tree, sub tree, index
```

Algorithm 5: initialization

```
Input: parents number
Output: pop
For I in the range parents number:
  New individual = spanning tree
  Add new individual into pop
Return pop
```

Algorithm 6: Crossover

Input: offspring, p1_index, p2_index
Output: new offspring1, new offspring2
p_1 = offspring[p1_index]
p_2 = offspring[p2_index]
position_1 = random(0, node_number(p_1))
position_2 = random(0, node_number(p_2))
P1 subtree = find subtree(p_1, position_1)
P2 subtree = find subtree(p_2, position_2)
new offspring1= replace(p_1, position_1, p2_subtree)
new offspring2= replace(p_1, position_1, p2_subtree)
Output: new offspring1, new offspring2

Algorithm 7: Mutation

Input: individual
Output: new individual
Tree = individual
position = random(0, node_number(tree))
New branch = spanning tree
New individual = replace(tree, position, new branch)
Return new indicidual

Exercise 5

This algorithm has 4 parameters, include population size, crossover probability, mutation probability and time budget. I chose three of the parameters to experiment to see which parameter settings will give the algorithm a better effect.

1. Population Size

The initial setting of crossover probability is 0.45, mutation probability is 0.4, time budget is 300s, the set of population size is 50, 100, 500, 1000. The result is shown as box plot in figure 1, it can be seen that when the population size is 500, the performance of the algorithm is best.

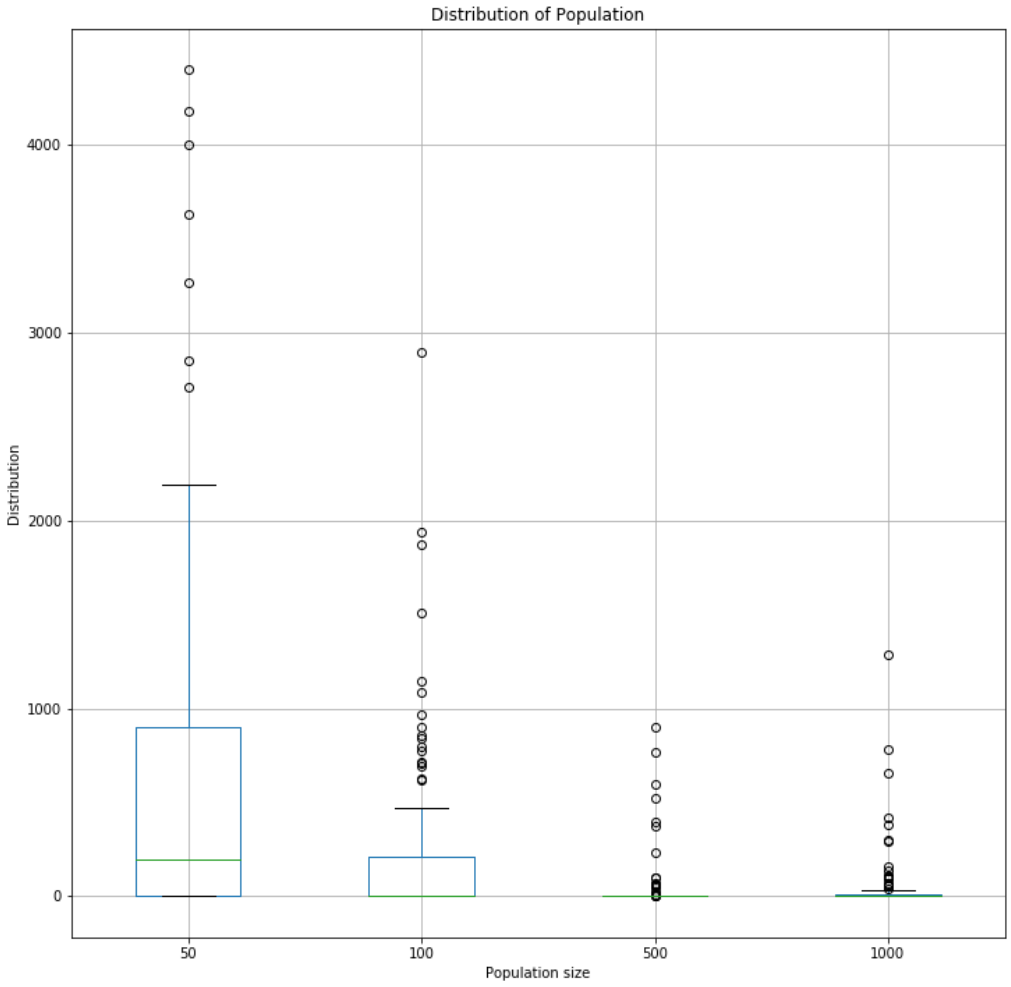

Figure 1 The distribution of population size

2. Crossover Probability

Choose the population size 500 which perform best in experiment 1, keep the set of mutation probability and time budget, change the set of crossover probability. The set of crossover probability is 0.25, 0.45, 0.65 and 0.85. The result is shown as box plot in figure 2, it can be seen that when the crossover probability is 0.45, the performance of the algorithm is best.

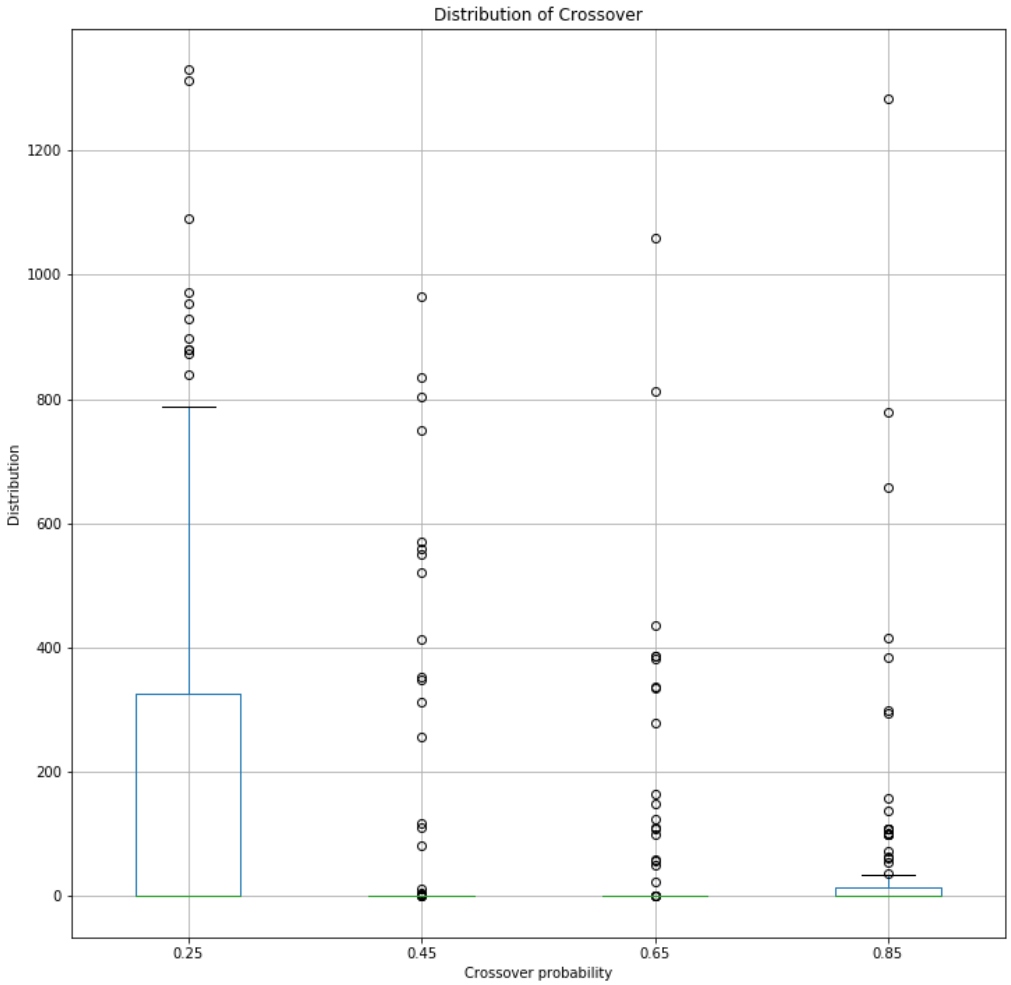

Figure 2 The distribution of crossover probability

3. Mutation probability

After the first and second experiments, when the set of population size is 500 and the set of crossover probability is 0.45, the performance of the algorithm is better than other settings. Using these two setting and keep the time budget, change the set of mutation probability. The set of mutation probability is 0.2, 0.4, 0.6, 0.8 and 0.99. The result is shown as box plot in figure 2, it can be seen that when the crossover probability is 0.2, the performance of the algorithm is best.

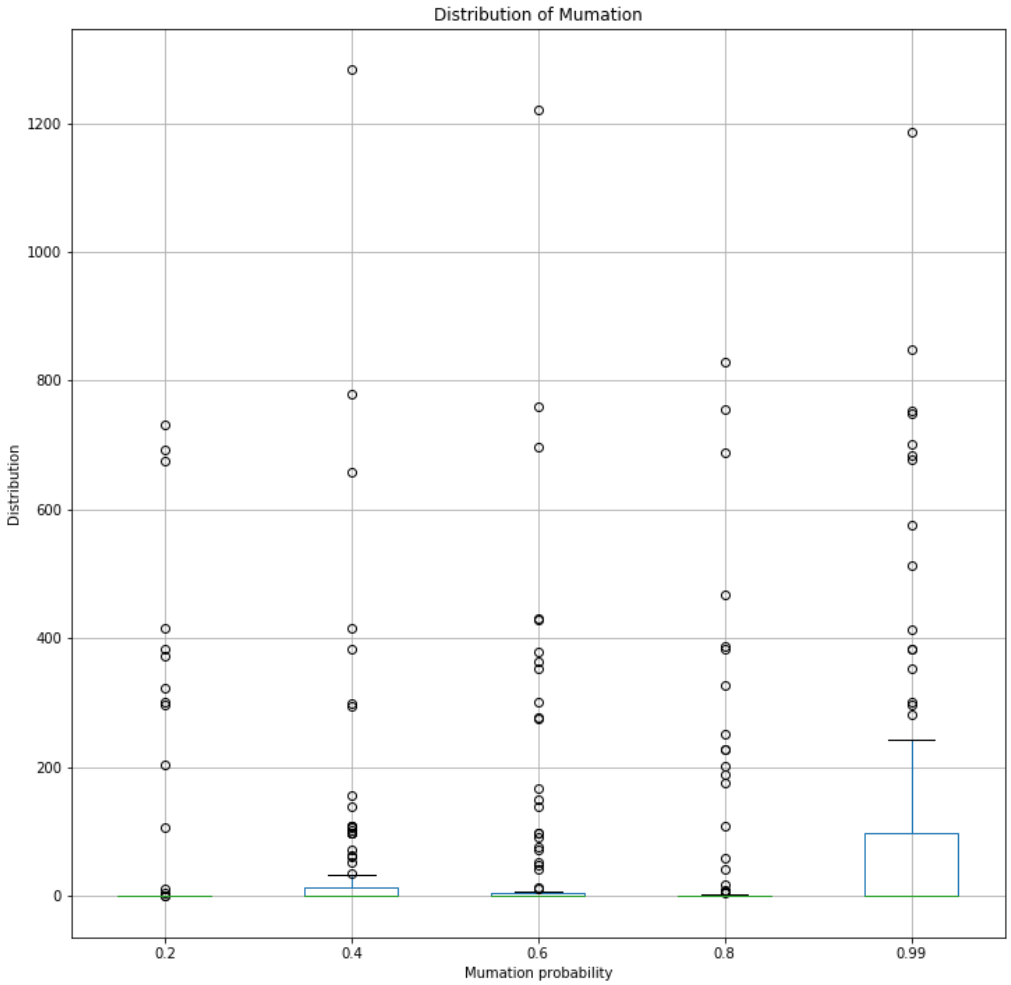

Figure 3 The distribution of mutation probability