Lab 5 EI Farol Bar Problem

Student Name: Binjie Zhang Student ID: 1870958

Exercise 4

Algorithm 1: Coevolution

```
Input: pop size, h, weeks, max_t
Output: result
Pop = initial (pop size)
Fit, flag, attendance = fitness
For iteration in range(max_t):
  For w in range(weeks):
    For I in range(len(pop)):
      Next_ds = strategy
      d_s.append(next_ds)
      attendance_situation = d
      state = s
      new_fit, flag, attendance = fitness(attendance_situation)
      Record result
    m = 0, s = 0, count = 0
    For I in range(len(fit)):
      If fit[i] > fit[s]: m = i
      Else if fit[i] > fit[s]: s = i
      Count = count + fit[i]
    Parent1, parent2 = pop[m], pop[s]
    P1, p2, a1,a2,b1,b2 = getm
    P1, a1, b1 = mutation
    P2, a2, b2 = mutation
    Pop = crossover
Return result
```

Algorithm 2: initial

```
Input: pop size, h
Output: pop
For I in range (pop size):
  Add h to individual
  For I in range(h):
    Add random number in individual
    For I in range(h):
      a = random dirichlet
    Extend individual with a
    For I in range(h):
      b = random dirichlet
    Extend individual with b
  Pop.append(individuals)
  Return pop
```

Algorithm 3: fitness

```
Input: result
Output: fit, flag, attendance
Fit = []
L = len(result)
Attendance = result.count(1)
If l == 0:   crowded = 0
Else: crowded = attendance/l
If crowded < 0.6:
  For I in range(l):
    If result[i] == 1: fit.append(1)
    Else: fit.append(0)
  Flag = 0
else
  For I in range(l):
    If result[i] == 0: fit.append(1)
    Else: fit.append(0)
  Flag = 1
Return fit, flag, attendance
```

Algorithm 4: mutation

```
Input: h, p, a, b
Output: p, a,b
Prob = random number
If prob > mutation prob:
  For I in range(h):
    p.append(random)
index_a = int(random* h)
Prob_a = random number
If prob_a > mutation prob:
  a = random dirichlet
index_b = int(random* h)
Prob_b = random number
If prob_b > mutation prob:
  b = random dirichlet
Retuen: p, a, b
```

Algorithm 5: crossover

```
Input: pop size, h, p1, p2, a1, a2, b1, b2
Output: new pop
For I in range (pop size):
  For j in range(len(p1)):
    Prob = random number
    If prob > crossover prob: p1[j], p2[j] = p2[j], p1[j]
```

```
   For j in range(0, len(a1), h):
      Prob = random number
      If prob > crossover prob: a1[j:j + h], a2[j:j + h] = a2[j:j + h], a1[j:j + h]
   For j in range (0, len(b1), h):
      Prob = random number
      If prob > crossover prob: b1[j:j + h], b2[j:j + h] = b2[j:j + h], b1[j:j + h]
   new1.append(h); new2.append(h)
   For I in range(h):
      new1.append(p1[i])
      new2.append(p2[i])
      new1.extend(copy (a1[i * h:(i * h) + h]))
      new2.extend(copy (a2[i * h:(i * h) + h]))
      new1.extend(copy (b1[i * h:(i * h) + h]))
      new2.extend(copy (b2[i * h:(i * h) + h]))
Return new pop
```

Exercise 5

The aim of co-evolutionary algorithm which leads the population to as efficient utilisation of the bar as possible (i.e., close to 60 %), while still not being overcrowded. So can use the number of attendances to judge how good the algorithm works. I chose three parameters to turning.

1. Population Size

The initial setting of the crossover probability is 0.45, the mutation probability is 0.45, the number of states in the strategies is 5, the number of weeks to simulate per generation is 100, the number of generations to run the simulation is 100, the set of population size is 10, 50, 100, 500, 1000. The result is shown as box plot in figure 1, it can be seen that when the population size is 1000, the performance of the algorithm is best.

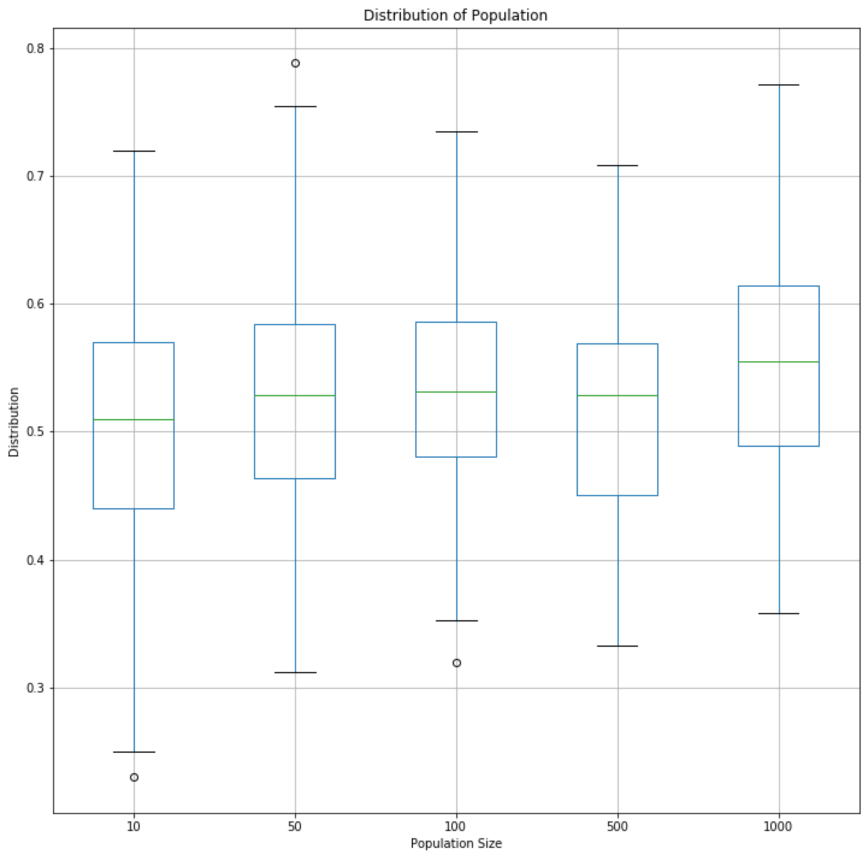

Figure 1. The distribution of population size

2. Crossover Probability

Choose the population size 1000 which perform, keep the other set, change the set of crossover probability. The set of crossover probability is 0.25, 0.45, 0.65, 0.85 and 0.99. 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. But the change in crossover has little effect on the results.

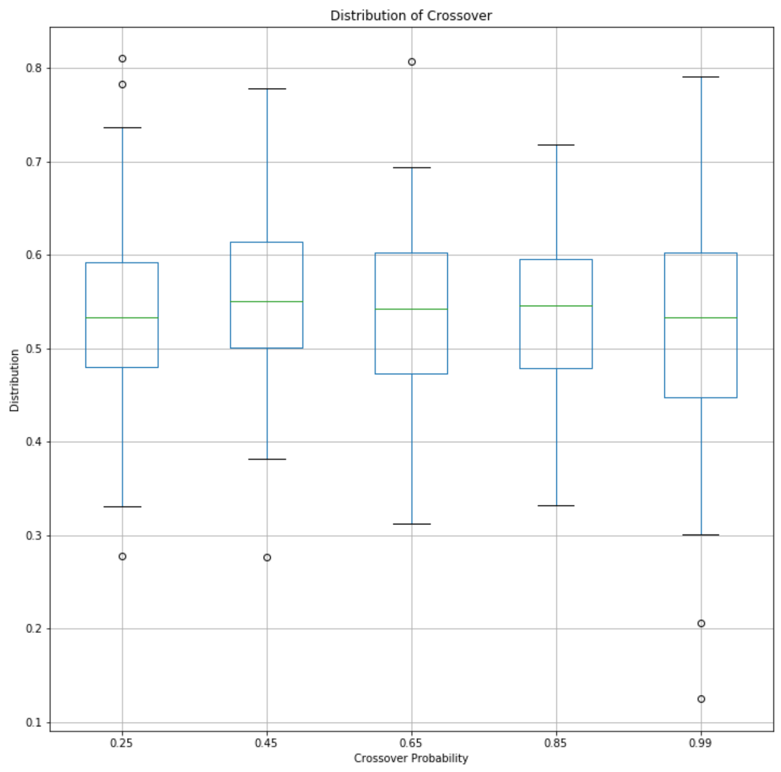

Figure 2. The distribution of Crossover probability

3. Mutation probability

After the first and second experiments, when the set of population size is 100 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 other setting, change the set of mutation probability. The set of mutation probability is 0.25, 0.45, 0.65, 0.85 and 0.99. The result is shown as box plot in figure 3.

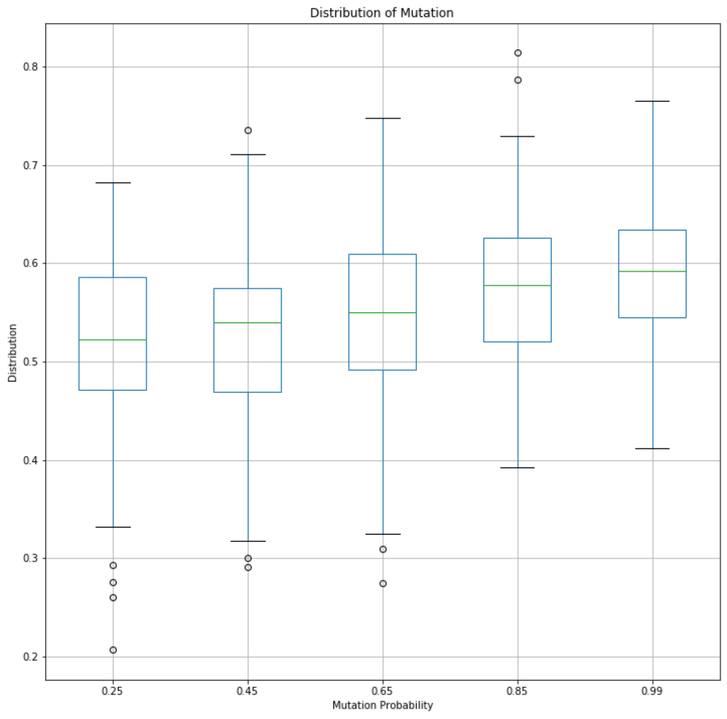

Figure 3. The distribution of Mutation probability