Antenna placement problem

DECEMBER 17

Saeed77t@gmail.com / student number: 40160957
Authored by: Reza Tahmasebi

Table of Contents

.

About the antenna placement problem

in this project, the goal is to find the best place and the best type of antennas and put them in such an order that inside an m*n area makes the best coverage with less cost.
To solve this problem, the genetic algorithm is used.

Report the code used in the algorithm

Parameters setup:

The width and height of the area are shown by *m* and *n, the* maximum amount of antennas is defined by *s,* and *t* is defining count of different types of antennas.

Initialize antennas types

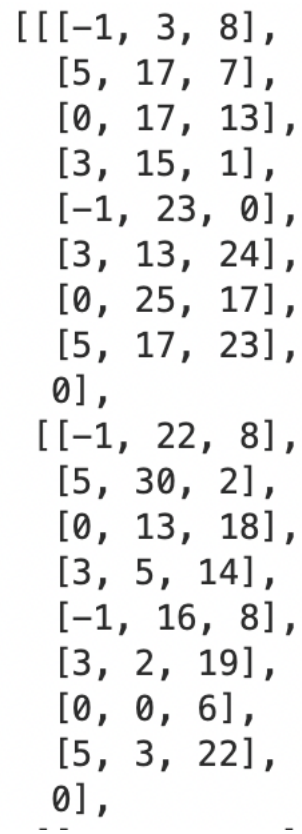
The antenna types differ randomly, and the cost of antennas has a linear relation with the radius of antennas. The antenna type is in the picture below.

```
[{'type': 0, 'cost': 22, 'radius': 8},
 {'type': 1, 'cost': 17, 'radius': 5},
 {'type': 2, 'cost': 24, 'radius': 9},
 {'type': 3, 'cost': 18, 'radius': 4},
 {'type': 4, 'cost': 12, 'radius': 3}]
```

Picture1: Antennas type

InitialPopulation function

This function gets the population size, antennas list, s, m, and n as the arguments, and it creates the first chromosomes with the number of population size.
Type -1 in the antennas list shows that the antenna is now installed, so it doesn't cover any area.
The chromosomes example is in the picture below.



Picture2: initial chromosomes example

The zero as the last chromosome member is a place to put the fitness number init.

ClacFitness function:

This function gets the population, s, antennas list, m, and n as the arguments and calculates the fitness based on the formula below,

$$f(x) = u(x) - \sum_{j}^{p(x)} c_t(j)$$

This function also checks whether antennas overlap and doesn't count the overlapped points.

FitSort function :

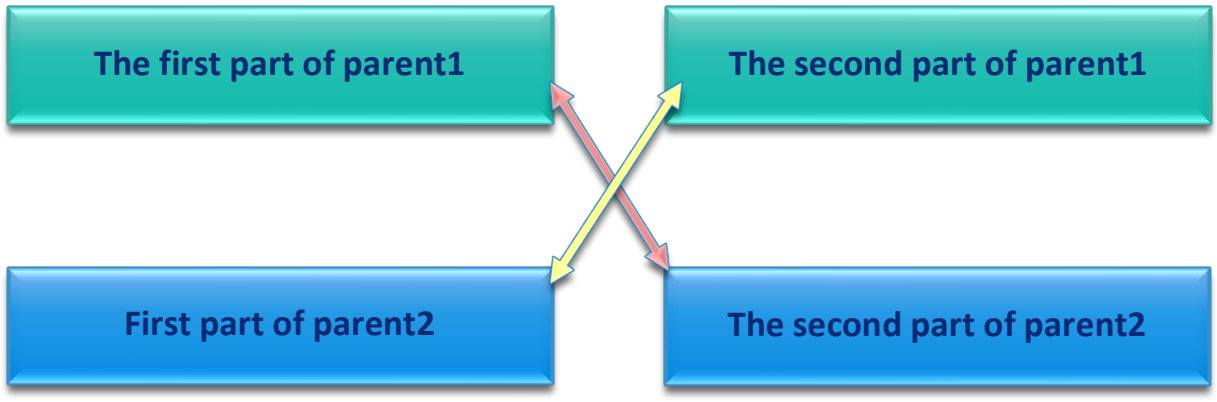
The function gets the population and s as arguments, sorting the chromosomes from best to worst based on their fitness. Fitness is the last member of each chromosome.

ParentSelection function :

This function gets the population and s as the arguments. It randomly chooses six members of the people, gives each of those members a roulette share based on their fitness, and then makes a tournament between them to determine a parent. This action will be repeated to select two parents from the population.

XoverBox function :

This function gets parents, antennas type, s, m, and n as arguments and selects a random break point. And it will break each parent into two parts; it consists of part one of the first parent and part two of the second parent together to make a new child, and also it consists second part of the first parent and the first part of the second parent together to make another child.



In the end, the function returns two children.

MutationBox function:

With a five percent mutation rate, this function gets children, s, antennas type, m, and n as arguments. And randomly chooses one of the triples and changes one of them, and this function returns the children.

ParentSelection2 function :

This function gets population, antenna type, m, and n as arguments. 6 members will be chosen randomly. After that, the process divides the width of the area (m) into some bins. Each of those members gets a roulette share based on how many bins they trigger. The more containers they start, the more share they have.
The roulette wheel starts, and a parent selects; this action happens two times to choose two parents.

Elitism function :

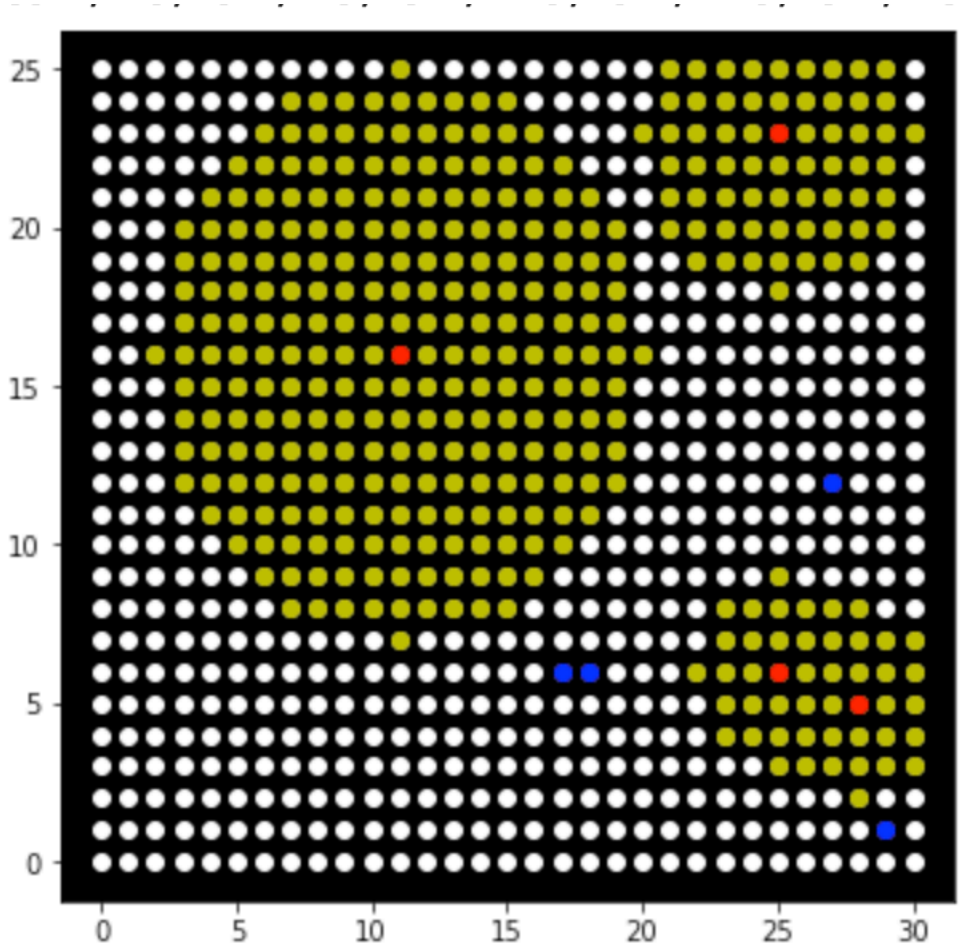
This function gets children and population as the arguments and will get the best parent based on fitness and replace them with the worst offspring. (1% elitism)

SurvivalSelection function :

This function gets children and population as arguments, removing half of the people and putting the children instead of the excluded population.

PlotResult function :

This function gets a chromosome, antenna type, s, m, and n, as arguments and will plot the area and antennas and the area covered by antennas.
The area points will show in white color, antennas with coverage will show red color, type -1 antennas will show blue color, and the covered area will be led by yellow color.



Picture3: example of plotting antennas

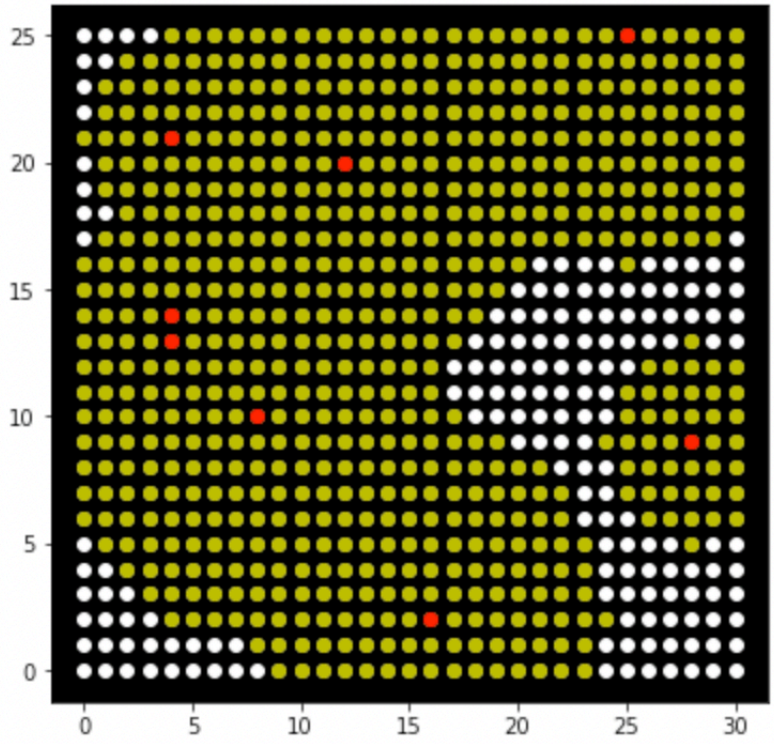
Survey on different setups

As the question asked, four setups have been implemented and will be checked in this report.

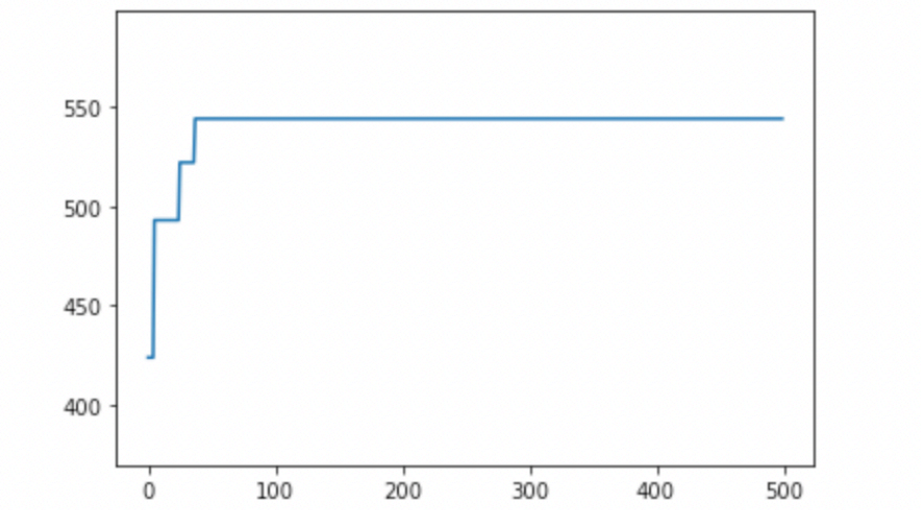
Setup one, as asked in the question.

This setup works with the *ParentSelection function* (tournament).
It uses the *XoverBox function* for one-point cross-over.
It uses the *MutationBox* function to mutate at a rate of 5%.
It uses the *elitism* function for survival selection. (1% elitism)
This set up executed 500 times, and the results are as follows:

The best chromosome: [[3, 8, 10],[1, 16, 2], [2, 4, 13], [3, 12, 20],[3, 25, 25], [4, 28, 9], [0, 4, 14],[4, 4, 21], 544]
The best fitness: 544
The answers plot :



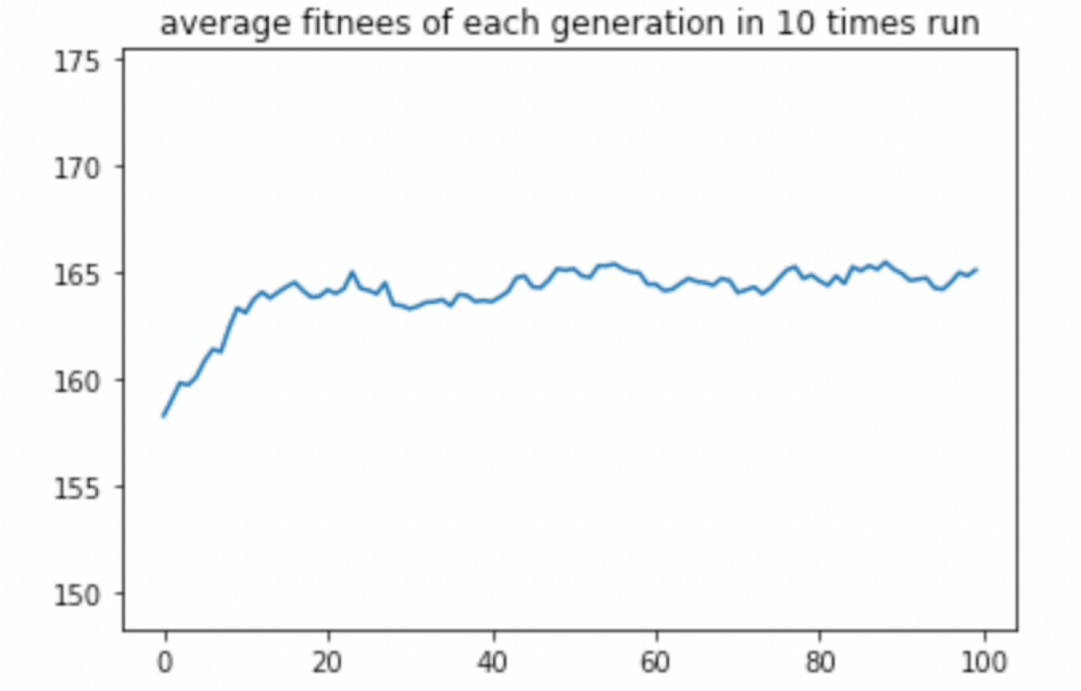
The answers fitness function :



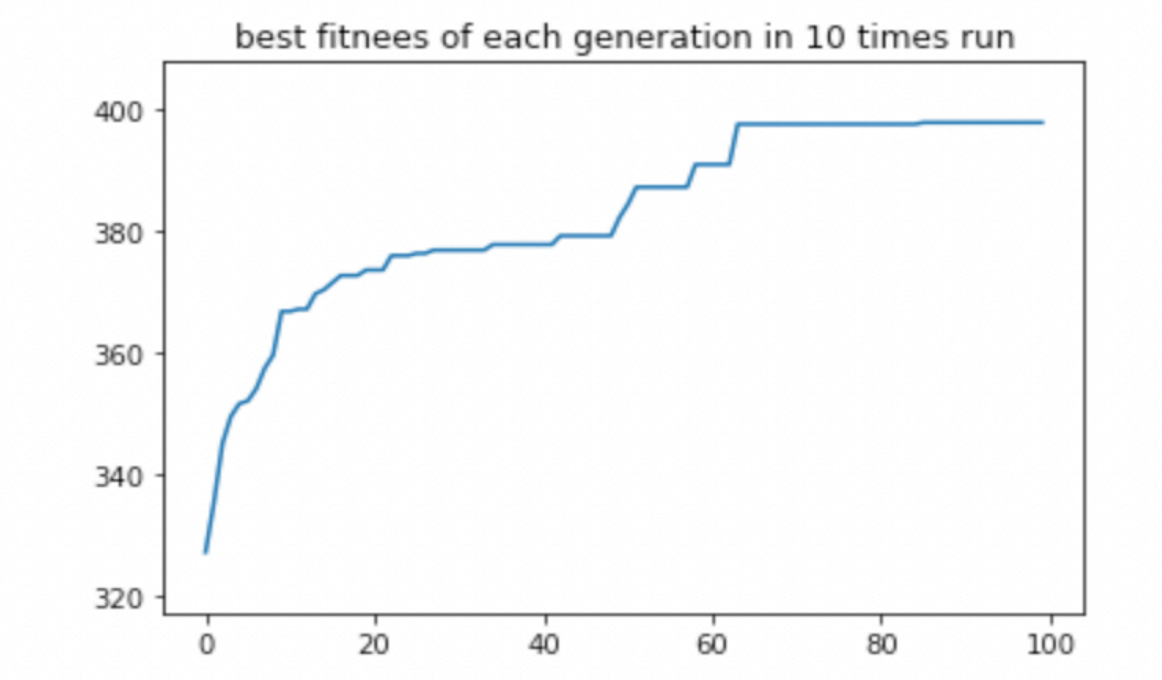
Run setup one for ten times individually:

After running the setup ten times, the results are as follows:

The average fitness function for each generation in times run:



The average best fitness of each generation after ten times runs:


best fitnees of each generation in 10 times run
400
380
360
340
320
0
20
40
60
80
100

Setup two, with different parent selections.

This setup works with the *ParentSelection2function* (tournament based on grids).

It uses the *XoverBox function* for one-point cross-over.

It uses the *MutationBox* function to mutate at a rate of 5%.

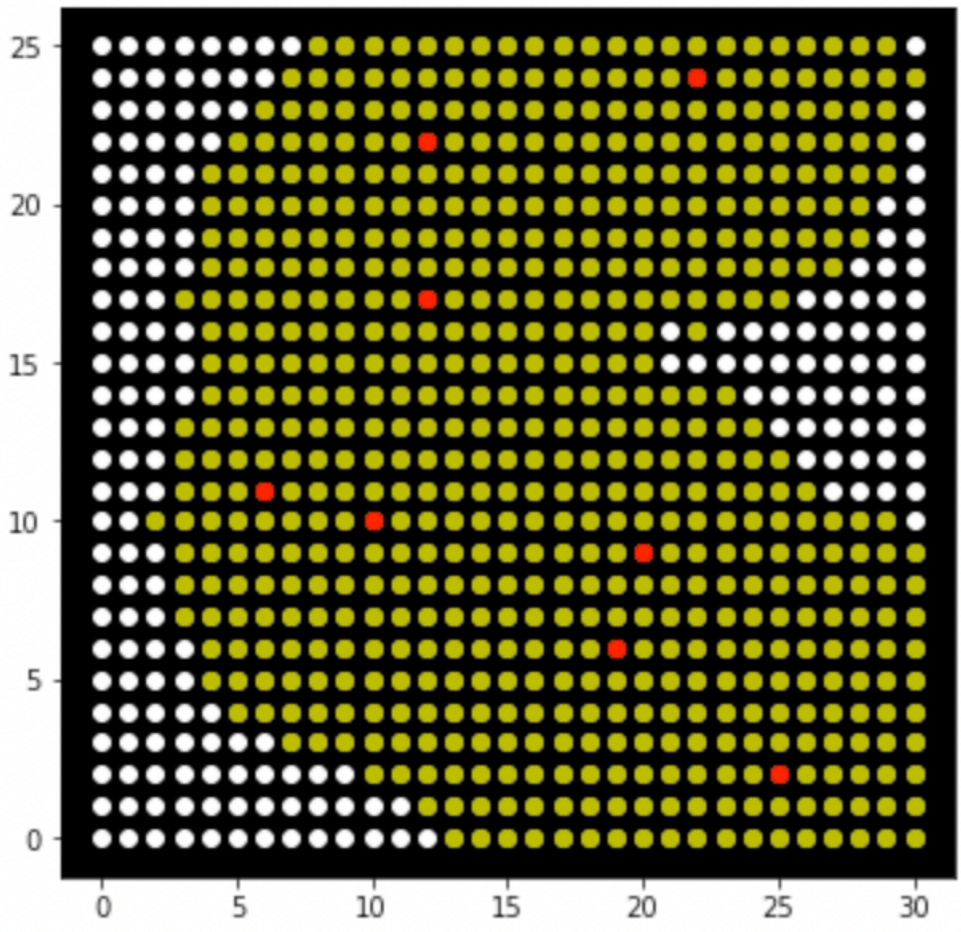
It uses the *elitism* function for survival selection. (1% elitism)

This set up executed 500 times, and the results are as follows:

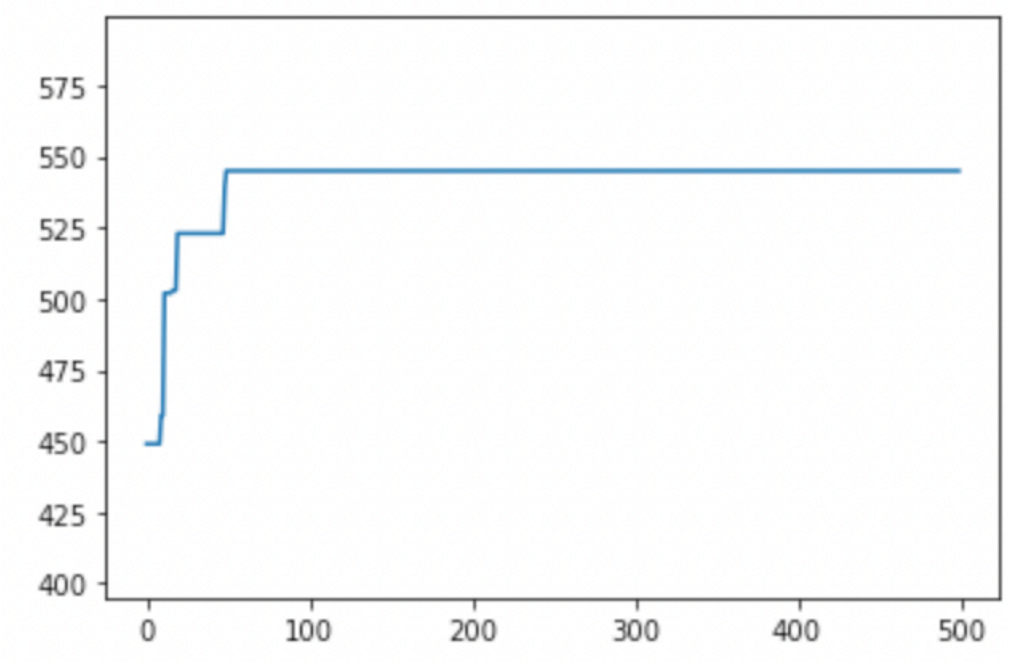
The best chromosome: [[0, 6, 11], [2, 20, 9],[3, 19, 6],[3, 25, 2],[3, 12, 17],[0, 12, 22], [1, 22, 24], [1, 10, 10], 545]

The best fitness: 545

The answers plot :



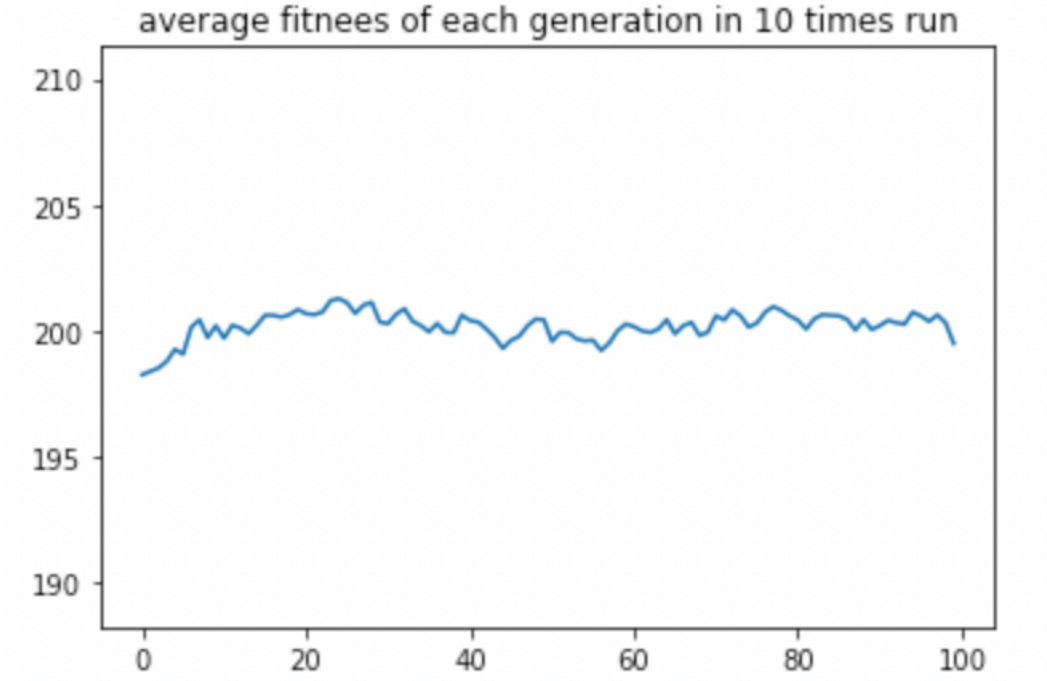
The answers fitness function :



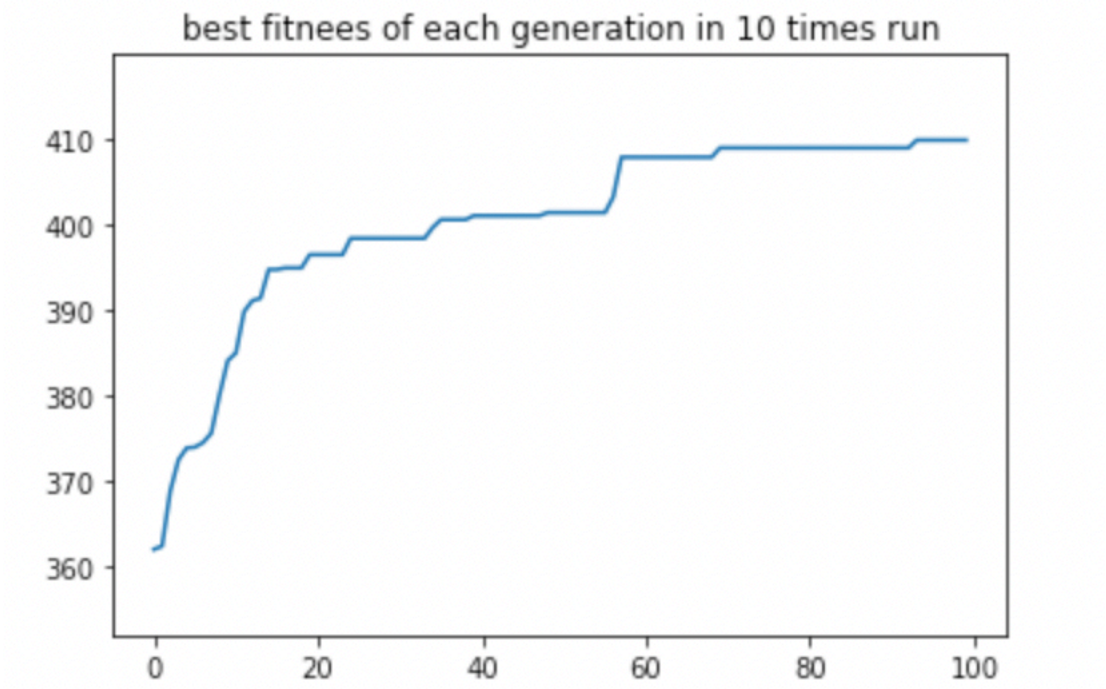
Run setup two for ten times individually:

After running the setup ten times, the results are as follows:

The average fitness function for each generation in times run:



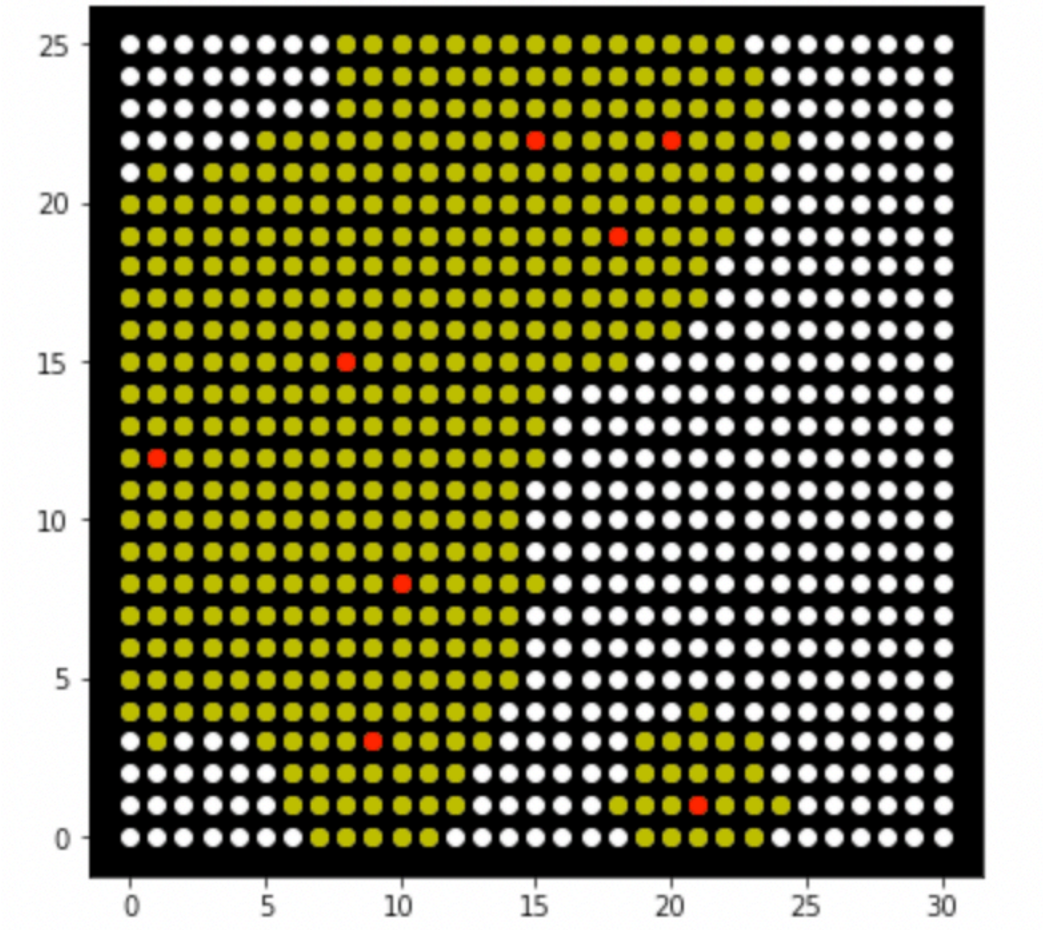
The average best fitness of each generation after ten times runs:



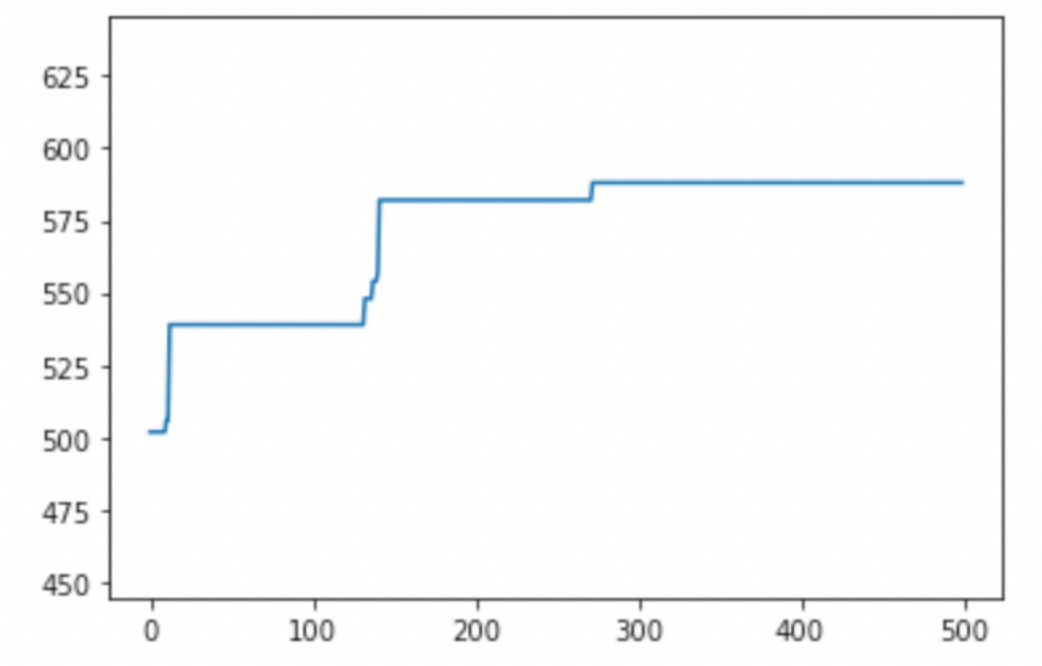
Setup three, with different survival selections.

This setup works with the *ParentSelection* (tournament).
It uses the *XoverBox function* for one-point cross-over.
It uses the *MutationBox* function to mutate at a rate of 5%.
It uses the *SurvivalSelection* function for survival selection. (remove half of the population)
This set up executed 500 times, and the results are as follows:

The best chromosome: [[3, 1, 12], [0, 21, 1], [2, 10, 8], [4, 20, 22], [0, 18, 19], [1, 8, 15], [4, 9, 3], [1, 15, 22],588]
The best fitness: 588
The answers plot :



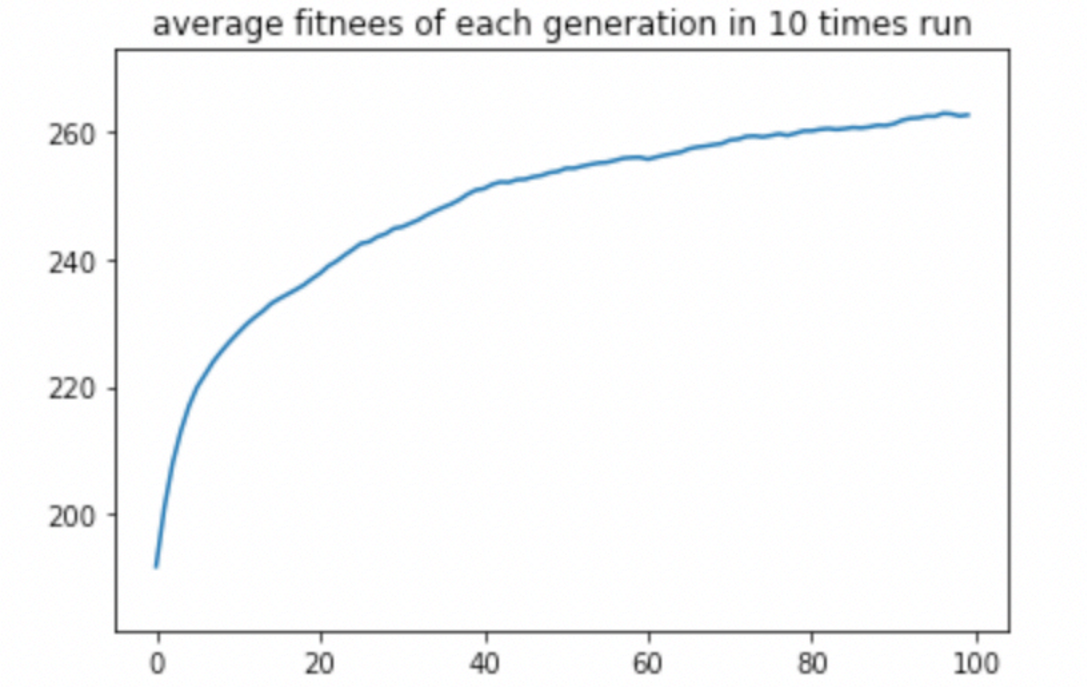
The answers fitness function :



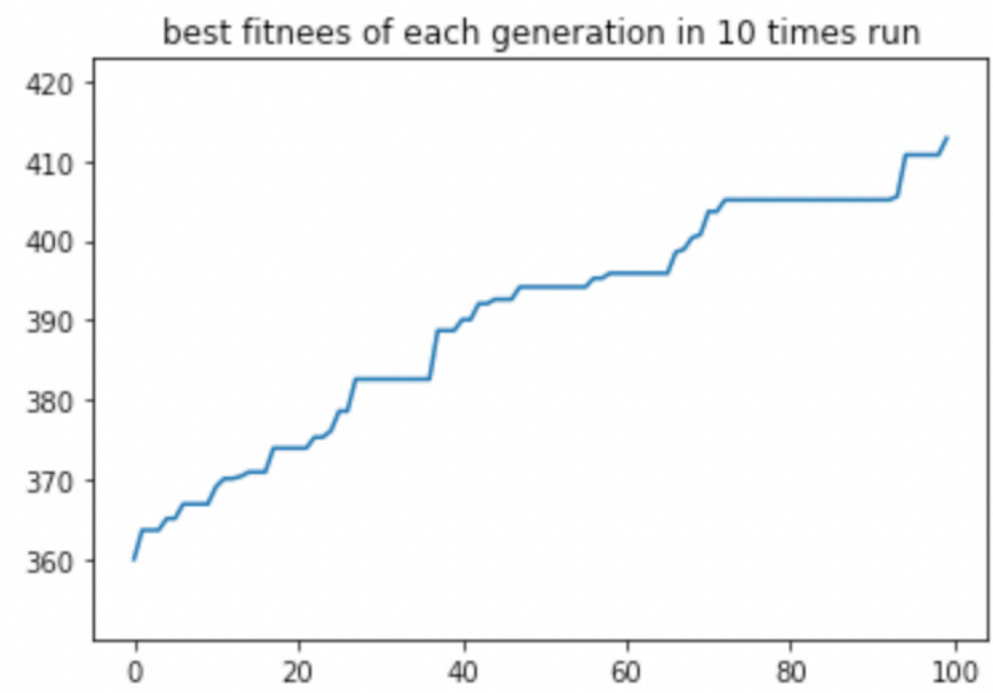
Run setup three or ten times individually:

After running the setup ten times, the results are as follows:

The average fitness function for each generation in times run:



The average best fitness of each generation after ten times runs:



Setup four, with different survival selections and parent selections.

This setup works with the *ParentSelection2function* (tournament based on grids).

It uses the *XoverBox function* for one-point cross-over.

It uses the *MutationBox* function to mutate at a rate of 5%.

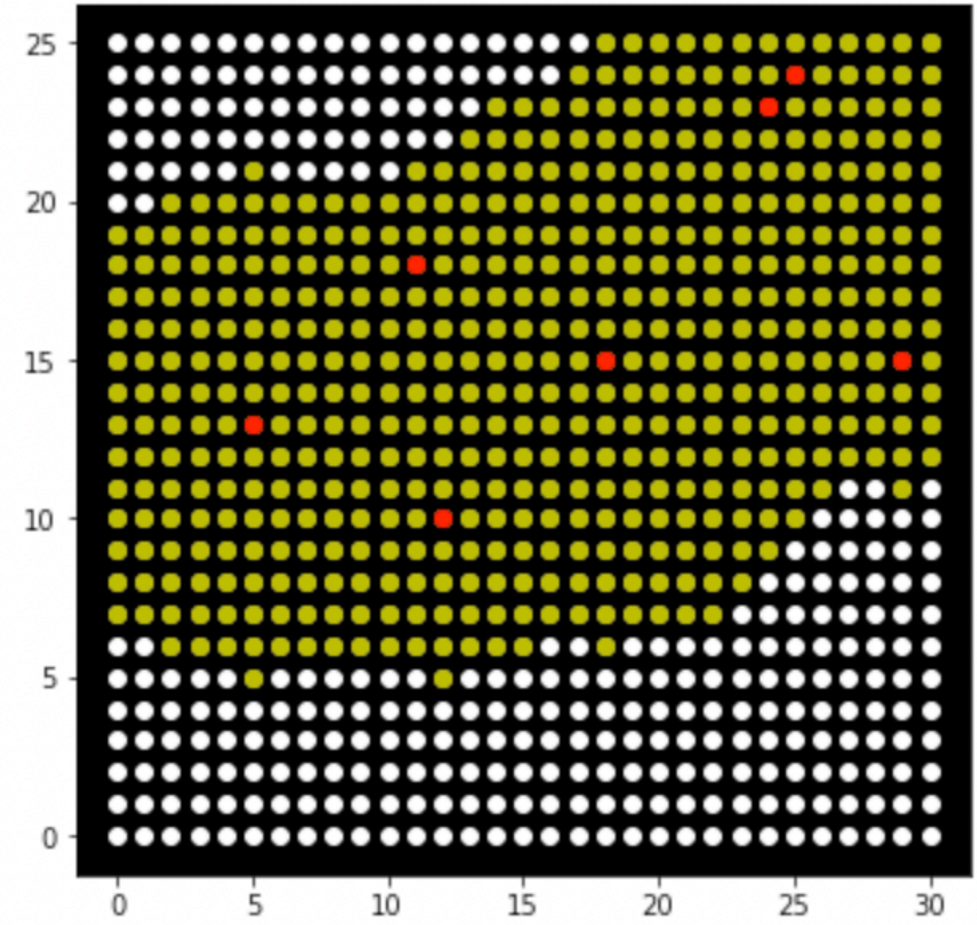
It uses the *SurvivalSelection* function for survival selection. (remove half of the population)

This set up executed 500 times, and the results are as follows:

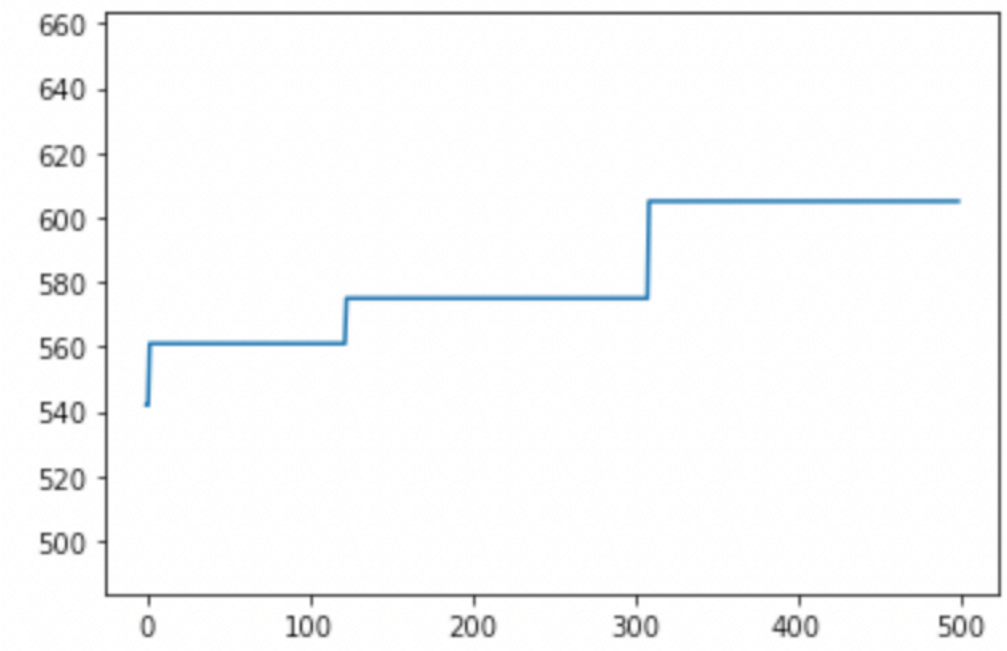
The best chromosome: [[2, 12, 10], [3, 18, 15], [1, 5, 13], [2, 24, 23], [4, 29, 15], [0, 11, 18], [2, 24, 23], [1, 25, 24], 605]

The best fitness: 605

The answers plot :



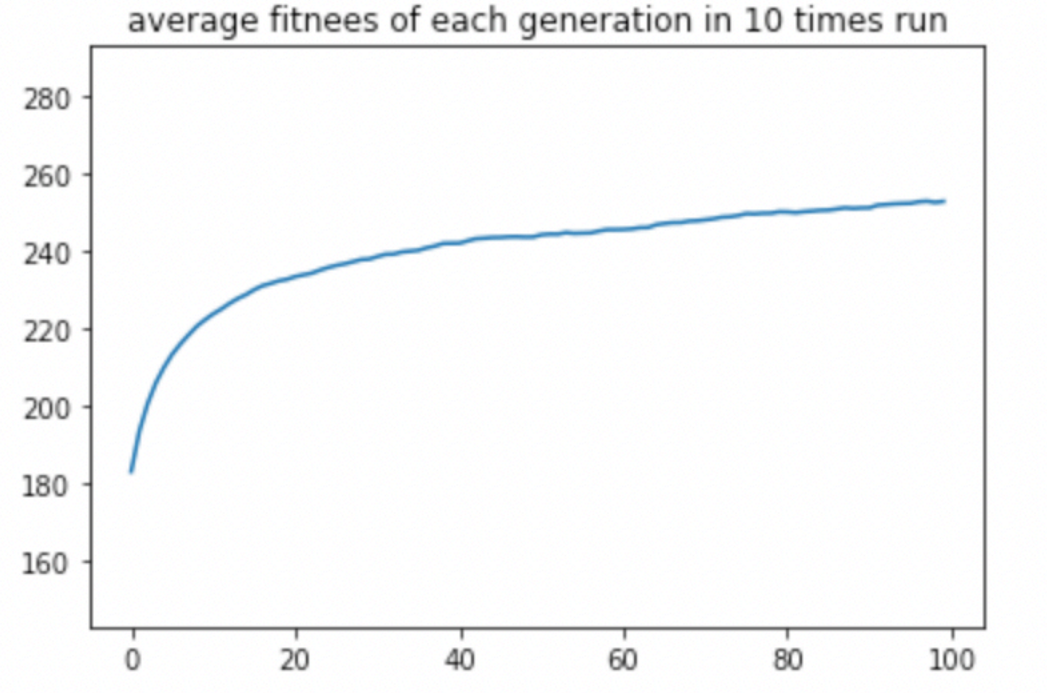
The answers fitness function :



Run setup four or ten times individually:

After running the setup ten times, the results are as follows:

The average fitness function for each generation in times run:



The average best fitness of each generation after ten times runs: