Evolutionary computation
A four-in-a-line game solver using a GP

General:
In this question, we have implemented strategies for four in a line game. This is a game for two players. There are balls of a certain color for each player, and each player in turn throws a ball of his color in the cells of the vertical board. The playing field consists of seven columns and six rows. The goal of the game is to collect four balls of your color vertically or horizontally or diagonally before the enemy does it.
Goal:
Find strategies (an algorithm) for winning the game with genetic programming.

Tools:
1) Programming language – Python.
2) Evolutionary computation framework – DEAP.

Genetic algorithm parameters:

Representation	Tree structures
Recombination	Single point crossover (exchange of subtrees)
Recombination probability	70%
Mutation	mutUniform (Random subtree replaced by the expression)
Mutation probability	0.1%

Parent selection	tournament
Population size	100 individuals
Initialization	random
Termination condition	100 generations

Process of work:

1. <u>Initialization</u>

Function set:

Function	Number of Input Terminals	Description
+	2	Addition Operator
-	2	Subtraction Operator
*	2	Multiplication Operator
%	2	Safe Division Operator
if_the_more_else	4	return x3 if x1>x2, else x4

Terminal set:

Terminal Value	Description
count1	number of "1" currently on the board
count2	number of "2" currently on the board
edges1	number of "1" on the edges
edges2	number of "2" on the edges
corner1	number of "1" on the corners
corner2	number of "2" on the corners
nearCorner1	number of "1" near the corner (the three squares around the corner)
nearCorner2	number of "2" near the corner (the three squares around the corner)
center1	is the "1" in the center
center2	is the "2" in the center
1, 3, 5, 7	Constants used to adjust weight

2. <u>Fitness Evaluation</u>

The next step is the evaluation of candidates. To evaluate the fitness parameter, we will create a function (evaluation()). The fitness of an individual is evaluated in a series of games, we used 500 games for each individual.
Where winning the game gives the player 1 point, and losing - 0 points. The fitness of an individual is the sum of the points he scored over 500 games.

3. Parent selection

The selection operator selects representatives for the use of genetic operators. For selection, we use *Tournament selection*. Selected candidates are then passed on to the next generation. In a 3-way tournament selection, we select 3-individuals and run a tournament among them. Only the fittest candidate amongst those selected candidates is chosen and is passed on to the next generation. In this way many such tournaments take place and we have our final selection of candidates who move on to the next generation.

4. Crossover

We use *one-point crossover*, in this crossover, a random crossover point in each individual is selected and and exchanged each subtree with the point as root between each individual. Resulting in two new representatives, with the genes of the first and second parent.

5. Mutation

The next step is the mutation operator (*mutUniform*). A point in the individual is randomly selected, then the subtree at that point as a root is replaced by the generated expression. Generation of expression also occurs as individuals in the first generation and with the same restrictions on the height of the tree.

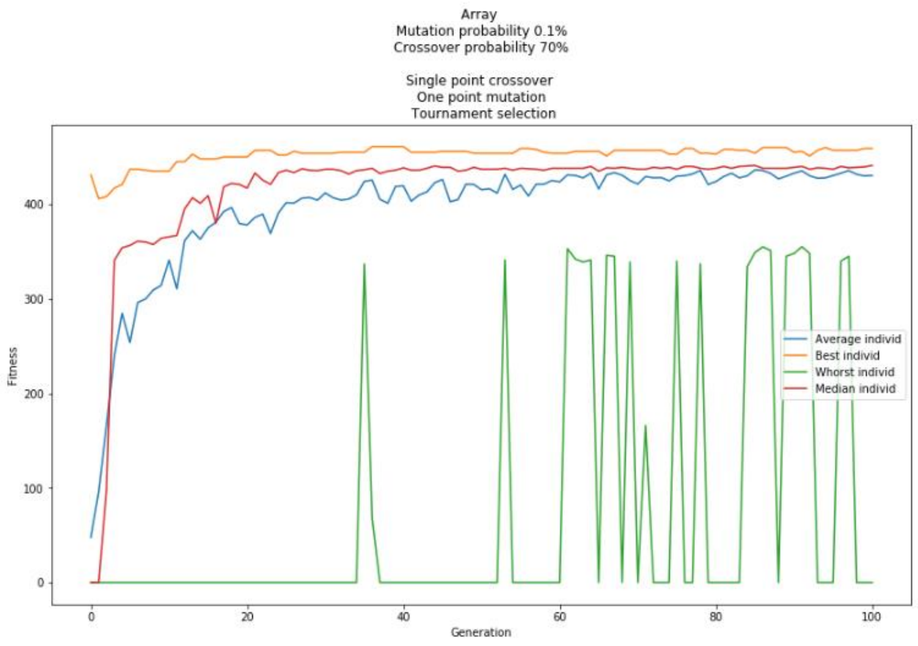
6. Final

The GP procedure is iterative. The final stage of each iteration is the formation of a new population. Stop occurs on the hundredth generation.

Results:

- During this experiment, several evolutionary runs were conducted using the above design with slight variations between runs. Below are the results the runs.

- We evaluated the genetic program by playing against a random player. In this case, the algorithm (strategy) was the first to play.

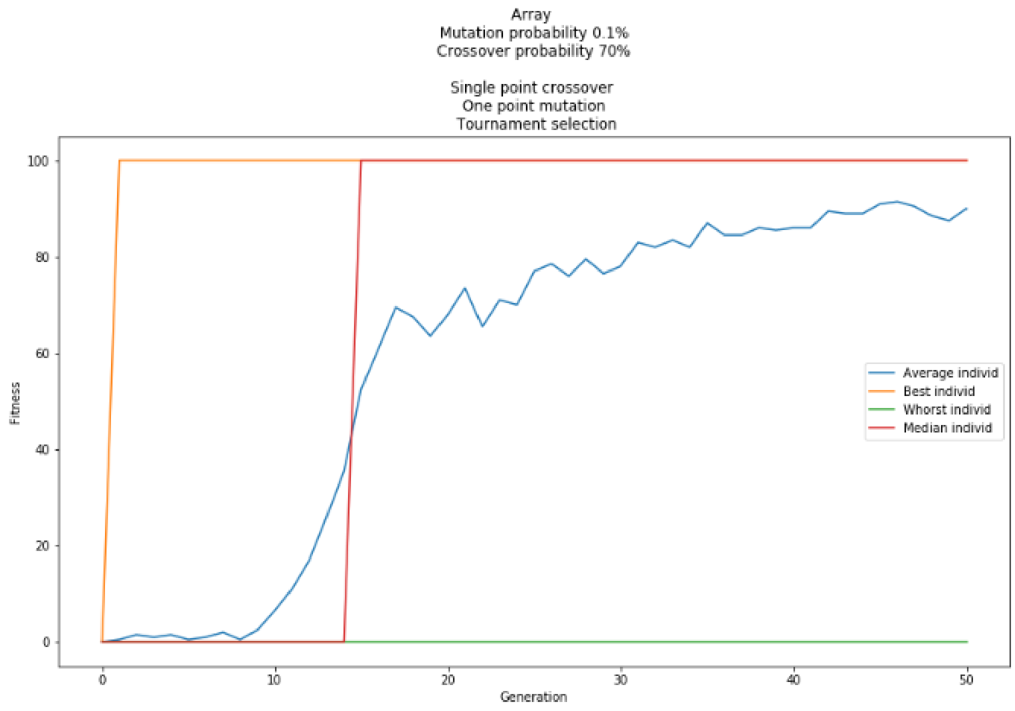


- Average time for termination: 1801s.
- The best player appears from the very beginning (in the first generation). The best player (for each generation) begins to win in 430 (out of 500) games starting around the 10th generation.
- Several interesting patterns were seen among the evolved players. Random players determine the next move by randomly choosing a move from among the set of current legal moves. The evolved players for this scenario tended to throw their balls directly on top of each other and start from the left column.
- As a result of modeling the evolution for the population, the best fitness value = 461, which corresponded to the following expression (game strategy):

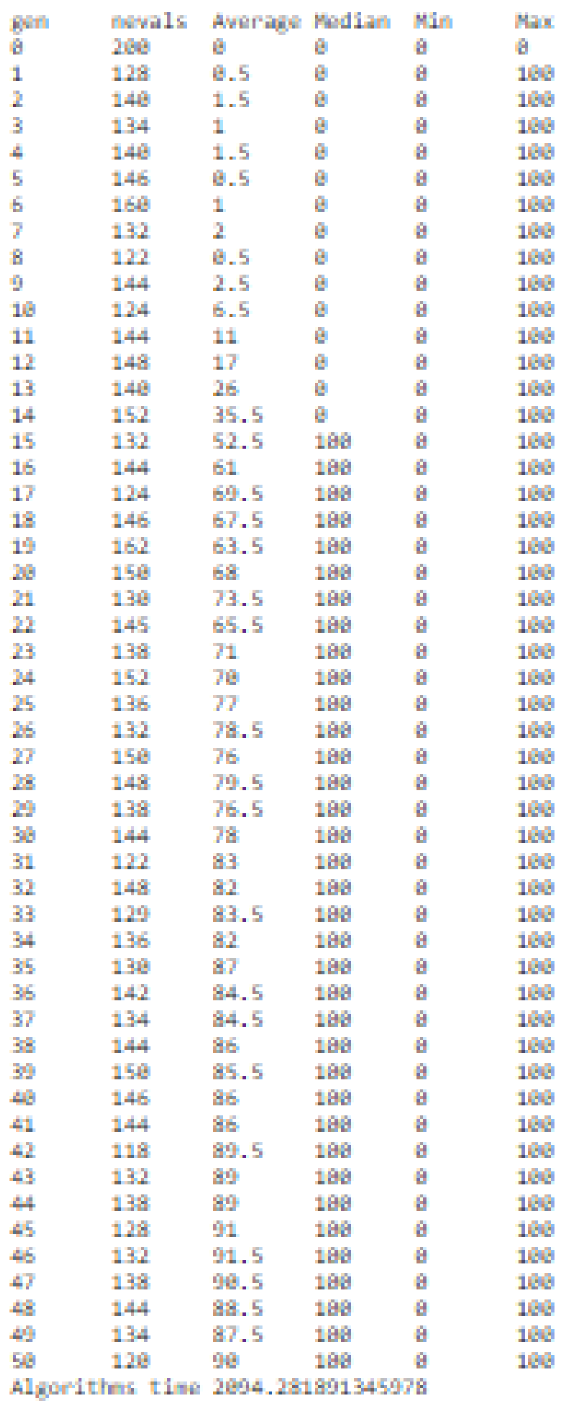
if_then_more_else(5, count2, 5, if_then_more_else(if_then_more_else(sub(corner1, count2), sub(if_then_more_else(count2, sub(5, count2), protectedDiv(corner1, center1), count2), 1), 1, if_then_more_else(1, count2, count2, 1)), sub(corner1, 5), 1, sub(sub(if_then_more_else(count2, if_then_more_else(sub(corner1, count2), sub(sub(if_then_more_else(5, sub(corner1, count2), protectedDiv(1, center1), count2), count2), 1), 1, if_then_more_else(1, count2, count2, 5)), protectedDiv(corner1, count2), count2), 1), 1)))

- The best player (the best tree) allowed us to get an attack strategy (the player makes the first move) in four in a line-game (a 86% win ratio).

- We took the 100 best strategies for the entire time of the first run. In the next run, we trained the GP algorithm to play against the selected 100 best strategies (trees). We trained for 50 generations (termination condition) and spent 100 games. In this case, the selected best players played first, and the strategy learned to respond to moves. So we got the defense strategy in four in a line-game.



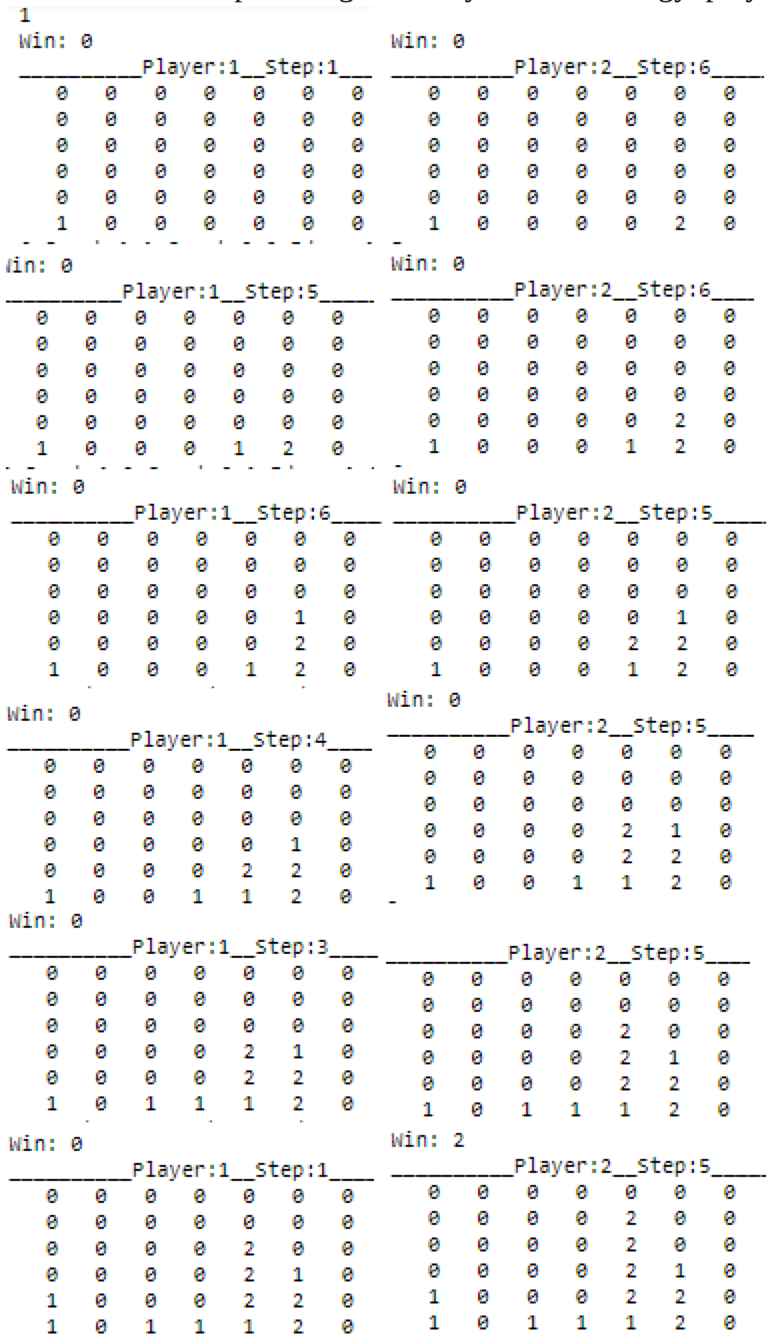
- Average time for termination: 2094s.
- A strategy learning against trees took longer than against a random player. Perhaps this is due to the fact that each tree requires additional time to calculate the terminals.
- We noticed good performance against the selected best players. Algorithm (strategy) won 100 out of 100 games (or 100% of the win).



```
gen     nevals  Average Median  Min     Max
0       200     0       0       0       0
1       128     0.5     0       0       100
2       140     1.5     0       0       100
3       134     1       0       0       100
4       140     1.5     0       0       100
5       146     0.5     0       0       100
6       160     1       0       0       100
7       132     2       0       0       100
8       122     0.5     0       0       100
9       144     2.5     0       0       100
10      124     6.5     0       0       100
11      144     11      0       0       100
12      148     17      0       0       100
13      140     26      0       0       100
14      152     35.5    0       0       100
15      132     52.5    100     0       100
16      144     61      100     0       100
17      124     69.5    100     0       100
18      146     67.5    100     0       100
19      162     63.5    100     0       100
20      150     68      100     0       100
21      130     73.5    100     0       100
22      145     65.5    100     0       100
23      138     71      100     0       100
24      152     70      100     0       100
25      136     77      100     0       100
26      132     78.5    100     0       100
27      150     76      100     0       100
28      148     79.5    100     0       100
29      138     76.5    100     0       100
30      144     78      100     0       100
31      122     83      100     0       100
32      148     82      100     0       100
33      129     83.5    100     0       100
34      136     82      100     0       100
35      130     87      100     0       100
36      142     84.5    100     0       100
37      134     84.5    100     0       100
38      144     86      100     0       100
39      150     85.5    100     0       100
40      146     86      100     0       100
41      144     86      100     0       100
42      118     89.5    100     0       100
43      132     89      100     0       100
44      138     89      100     0       100
45      128     91      100     0       100
46      132     91.5    100     0       100
47      138     90.5    100     0       100
48      144     88.5    100     0       100
49      134     87.5    100     0       100
50      120     90      100     0       100
Algorithms time 2094.281891345978
```

- As a result, we got two strategies: 1. when the algorithm goes first, 2. when the algorithm goes second.
- We also realized the opportunity for the best player to play against a person. We held a series of games the best player against the man. In this series of games, the best player showed himself poorly, and defeating him was not difficult.

- Below is an example of a game. Player 2 is a strategy, player 1 is a novice person:



```
 1
 Win: 0                               Win: 0
 __________Player:1__Step:1___        __________Player:2__Step:6____
    0   0   0   0   0   0   0            0   0   0   0   0   0   0
    0   0   0   0   0   0   0            0   0   0   0   0   0   0
    0   0   0   0   0   0   0            0   0   0   0   0   0   0
    0   0   0   0   0   0   0            0   0   0   0   0   0   0
    0   0   0   0   0   0   0            0   0   0   0   0   0   0
    1   0   0   0   0   0   0            1   0   0   0   0   2   0

Win: 0                                Win: 0
__________Player:1__Step:5____        __________Player:2__Step:6___
   0   0   0   0   0   0   0             0   0   0   0   0   0   0
   0   0   0   0   0   0   0             0   0   0   0   0   0   0
   0   0   0   0   0   0   0             0   0   0   0   0   0   0
   0   0   0   0   0   0   0             0   0   0   0   0   0   0
   0   0   0   0   0   0   0             0   0   0   0   0   2   0
   1   0   0   0   1   2   0             1   0   0   0   1   2   0

 Win: 0                               Win: 0
 __________Player:1__Step:6____       __________Player:2__Step:5____
    0   0   0   0   0   0   0            0   0   0   0   0   0   0
    0   0   0   0   0   0   0            0   0   0   0   0   0   0
    0   0   0   0   0   0   0            0   0   0   0   0   0   0
    0   0   0   0   0   1   0            0   0   0   0   0   1   0
    0   0   0   0   0   2   0            0   0   0   0   2   2   0
    1   0   0   0   1   2   0            1   0   0   0   1   2   0

                                      Win: 0
Win: 0                                __________Player:2__Step:5___
__________Player:1__Step:4___            0   0   0   0   0   0   0
   0   0   0   0   0   0   0             0   0   0   0   0   0   0
   0   0   0   0   0   0   0             0   0   0   0   0   0   0
   0   0   0   0   0   0   0             0   0   0   0   2   1   0
   0   0   0   0   0   1   0             0   0   0   0   2   2   0
   0   0   0   0   2   2   0             1   0   0   1   1   2   0
   1   0   0   1   1   2   0
Win: 0
__________Player:1__Step:3____        __________Player:2__Step:5___
   0   0   0   0   0   0   0             0   0   0   0   0   0   0
   0   0   0   0   0   0   0             0   0   0   0   0   0   0
   0   0   0   0   0   0   0             0   0   0   0   2   0   0
   0   0   0   0   2   1   0             0   0   0   0   2   1   0
   0   0   0   0   2   2   0             0   0   0   0   2   2   0
   1   0   1   1   1   2   0             1   0   1   1   1   2   0

Win: 0                                Win: 2
__________Player:1__Step:1___         __________Player:2__Step:5____
   0   0   0   0   0   0   0             0   0   0   0   0   0   0
   0   0   0   0   0   0   0             0   0   0   0   2   0   0
   0   0   0   0   2   0   0             0   0   0   0   2   0   0
   0   0   0   0   2   1   0             0   0   0   0   2   1   0
   1   0   0   0   2   2   0             1   0   0   0   2   2   0
   1   0   1   1   1   2   0             1   0   1   1   1   2   0
```

- As we can see from the example of the game presented above, the resulting trained strategy can defeat a person who is not very capable in the game.

Conclusions:

➢ From the fitness graphs in different generations, we see that with the help of genetic programming, we were able to develop players who can win the four in a line game.

- The functions and terminals selected and used in the experiment allow us to develop excellent players for playing four in a line.
- The strategy of the best player showed a high level of the game and really leads to victory if you follow it when playing against a random player or a novice person.
- We noticed that a winning strategy in most cases is a simple strategy - always throw the next ball in the same column. Thus, when playing against players who are not so smart, such a strategy will help to win.

Future Work:

- In this work, we were able to develop good players who could easily defeat the Random player. The player's skill level was not enough to be able to compete effectively against a human competitor. In future work, we propose to improve the player on this with more variations on the genetic program model.

Credits:

- DEAP documentation https://deap.readthedocs.io/en/master/
- Sheppard Clinton. Genetic Algorithms with Python
- Eiben A.E., Smith J.E. Introduction to Evolutionary Computing