Report

Github Link: https://github.com/maithrreye/Cant-stop-game

Evolutionary Algorithm:

EvolutionaryAlgorithm.py script has all the methods to synthesize script.

- The Init function creates 'n' number of scripts by selecting random rules and return the objects.
- Evaluate () function evaluates the script by allowing them to play as both test and random player and find their fitness value. This function also clear cache after evaluation and remove unused rules and recreate the scripts with used rules.
- elite () function selects best 'e' script based on its fitness value
- tournament selects best script from random 't' scripts.
- cross over mixes if clause of two scripts and generate new rule list which is then used by Mutation function to create a new script.
- This algorithm will remove duplicate rules in the rules set.
- This algorithm creates unique name to the script and also clear pycache (both .py and .pyc files) using Os and shutil package and recreates script based on updated rule set (after removing unused rules)
- At the end of last evaluation, it returns the name of a script that has largest Fitness value and its Fitness value.

Remove unused Rule:

After evaluation, it checks rules of each script and append the rules to the temporary list if the counter value for that rule is greater than 0. Now the temporary list is updated as new rule set for each script object. This removes the rules that was never used.

```
for i in range(len(self.Pscript)):
        rules=self.Pscript[i].getRules()
        lengthofRules=len(rules)
        obj=self.P[i]
        print(str(obj))
        countercalls=self.P[i].get_counter_calls()
        for j in range(0,lengthofCounter):
            r=rules[j]
            temp=countercalls[j]
            if temp > 0:
                new_rules.append(r)
          if len(new_rules) > 0:
            rules = new_rules
      setlist=set(rules)
      rules=list(setlist)
      self.Pscript[i].setRules(rules)
```

Experiment with Part II

The rule that always made test player to win is

```
if DSL.hasWonColumn(state,a) and DSL.isStopAction(a):
        return a
```

This strategy always makes the test player to win with more than 90% victory.

I ran three experiment to generate Script with high fitness value by setting different n, L and e value. Following are the result of Random player playing the script synthesized from EZS algorithm and test player playing the above mention strategy

Parameter	**Random Player (Player 1)** Script from EZS algorithm	**Test Player (Player 2)**
n=30, e=10, t=5, l=2	Script 11 – Loss (0)	Won (100)
n=20, e=5, t=5, l=3	Script 16 -Loss (2)	Won (98)
n=20, e=10, t=5, l=2	Script 0 – Loss (5)	Won (95)
n=20, e=10, t=10, l=3	Script 12- loss (10)	Won (90)

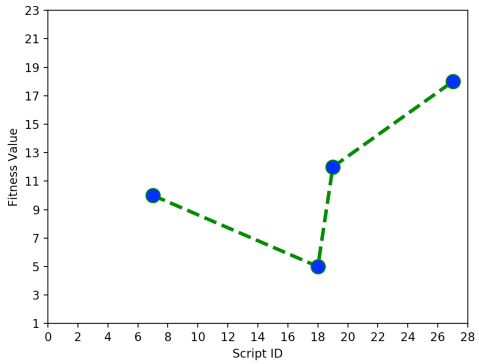
Though most of the script synthesized have the above-mentioned strong rule as part of their rule set, but it fails to make it even 50 percent success rate comparing to the script used by Test player. The strategy that I have written for the test player always outperforms the script synthesised by Evolutionary Algorithm.

Results and Discussion:

- This small version of Can't stop is too simple and hence it doesn't need too many strategies to make the player win. One Simple strategy is enough to make a player always win.
- I believe the script synthesised by EZS algorithm can win if the board size is win and the population is around 30-50 and the Generations are around 5-8. This is because strategies like 'DSL.numberPositionsProgressedThisRoundColumn(state, NUMBER) > SMALL_NUMBER and DSL.numberPositionsConquered(state, NUMBER) > SMALL_NUMBER' might be more helpful when the board size is big and each column has more number of steps
- One common pattern I found from the resultant highest fitness script is that they have "DSL.hasWonColumn(state,a) and DSL.isStopAction(a)" in their rule set that has maximum counter value comparing to the other rule's counter value in samescript. Also, another interesting observation is that, the combination of winning strategy with other strategy like "DSL.hasWonColumn(state,a) and DSL.isStopAction(a) and DSL.isDoubles(a)" in the same rule does not yield high counter value. This concludes that DSL.hasWonColumn(state,a) and DSL.isStopAction(a) is dominant strategy. You play just this strategy; you win all the time.

Experiment	Winning Script	Highest Counter value Rules	Lowest Counter value rules
n=30, e=10, t=5, l=2	Script 11	DSL.hasWonColumn(state,a) **and** DSL.isStopAction(a) **and** DSL.isStopAction(a): DSL.isStopAction(a) **and** DSL.hasWonColumn(state,a):	DSL.containsNumber(a, 4) **and** DSL.hasWonColumn(state,a) **and** DSL.containsNumber(a, 2): DSL.isDoubles(a) **and** DSL.isDoubles(a):
n=20, e=5, t=5, l=3	Script 16	DSL.isStopAction(a) **and** DSL.hasWonColumn(state,a) DSL.numberPositionsProgressedThisRoundColumn(state, 6) > 1 **and** DSL.isStopAction(a)	DSL.isDoubles(a) **and** DSL.containsNumber(a, 5) **and** DSL.isDoubles(a) DSL.isDoubles(a) **and** DSL.isDoubles(a)
n=20, e=10, t=10, l=3	Script 12	DSL.isStopAction(a) **and** DSL.hasWonColumn(state,a) **and** DSL.hasWonColumn(state,a) DSL.isStopAction(a)	DSL.isDoubles(a) **and** DSL.hasWonColumn(state,a) DSL.containsNumber(a, 2) **and** DSL.containsNumber(a, 5) **and** DSL.actionWinsColumn(state,a)

- For a small size population ‘n’ around 15 with ‘e’ 5 and ‘l’ 2, I could see most of the game finishes with match draw and there is high negative value. At the end of this synthesis only 4 script had positive high fitness value.



- One observation with the script given with random and test player is that the test player is able to win the game, but he loses the point when he doesn’t stop the game whenever needed (isdouble (a) as example in test script). This kind of gave an understanding how important is it to decide when to stop in the Can’t stop game. Further with this, I was able to see rules that has “isstop(a)” in their sub-rule had a decent countervalue.

- I would argue that the script synthesised by EZS algorithm is not really helping here in a small version of Can’t stop if the population is around 30. The strategy is simple in this case. But I would still believe this can be helpful for a bigger version of can’t stop and also when synthesizing from 80-100 script and having two different script pool for each player instead of using same script pool for both the players.

- Key takeaway from this assignment:- It expanded our thought process on how to derive strategy and expand them, how strategies influence in deciding the winner, how genetic algorithms can be used to select good strategy and how we can use different approaches in synthesizing script to use it for different other games.