Report

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

Evolutionary Algorithm:

Init Function:

- Create Rule set Randomly from the DSL pool.
- Creates initial 'pop_n' number of script and written Script object that would access Id, rules from Script.py
- The rule set will have unique combination of rules

Evaluate Function:

- Instantiating object for the script created by Init Function.
- This function allows each script to play as both Player 1 and Player 2.
- It gives 1 if a player wins a match and 0 if player losses the match and a -1 if the game is draw.
- Win and loss score are maintained by separate array and Final Fitness score is calculated from No of victories and no of losses.
- Remove the unused Rules
- Clear py cache using OS library (Both .py and .pyc) files
- Recreate script with unique name and hence no two script will have same name.

Elite Function:

- Returns 'e' scripts that has highest fitness value

Tournament Function:

- Randomly chooses 't' index value and selects 2 best script from that index based on the highest fitness value.

Crossover:

- Crossover gets two script from Tournament and split the Ruleset of each script and mix it to become a child script.
- This new ruleset is given to Mutation.

Mutation:

- Generate random rules and add it before or after the rules in the child ruleset

Rest of EZS algorithm:

- Create new script from ruleset returned by Mutation and update the P'. it uses unique Ids for the script and hence no two script will have same name.
- Once the number of populations of P' is equal to 'n' population, it updates P that has Instance of script with new script instance + instance of best script from elite.
- 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, the removed unused rule function checks rules of each script and append the rules to the temporary list if the counter value for that rule is greater than 0.
- The temporary list is updated as new rule set for each script object.
- This removes the rules that was never used, and updated ruleset is used to recreate the script after clearing the pycache.

```
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)
```

<u>Experiment with Part II</u>

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)
n=20, e=5, t=5, l=3	Script 58 – loss (0)	Won (100)
N=15 e=10 t=10 l=2	Script8- loss(11)	Won(89)

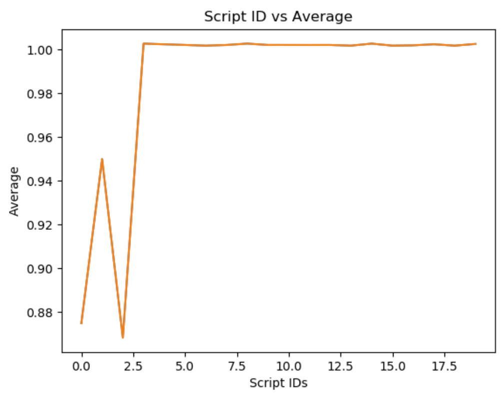
Though most of the script synthesized have the above-mentioned strong rule as part of their rule set, but it fails to make it even 50 percent success rate comparing to the script used by Test player. The strategy that I have written for the test player always outperforms the script synthesised by Evolutionary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

Also below is the result playing with Random script vs Script synthesised by EZS algorithm. Since Player 1 is playing random action, each script is run thrice to see if the Synthesized script won or loss. Only Script 8 and Script 16 synthesised by the EZS algorithm win against the Random player.

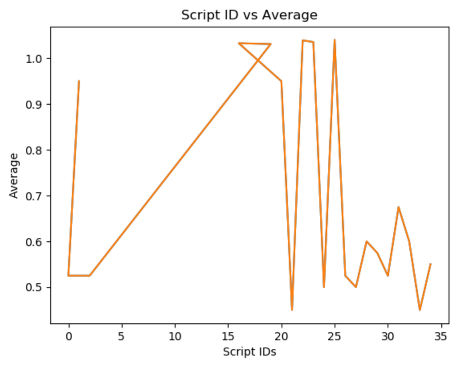
Parameter	**Random Script Player 1**	**Script by EZS player 2**
n=15 e=10 t=10 l=2	22 (loss)	Script 8 -78 (won)
n=20, e=5, t=5, l=3	100 (win)	Script 58 -0(loss)
n=20, e=10, t=10, l=3	95 (win)	Script 12- 5 (loss)
n=20, e=5, t=5, l=3	49 (loss)	Script 16 – 51 (win)
n=30, e=10, t=5, l=2	100(win)	Script 11- 0 (loss)

Results and Discussion:

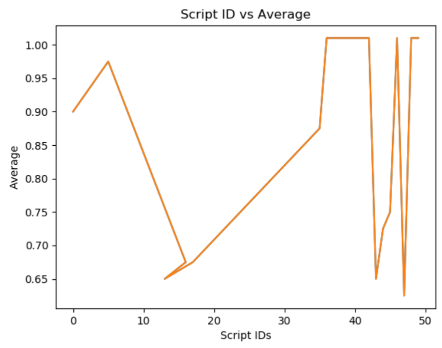
The following is the Average victory value of a script. The Experiment is run for n=20, e=5, t=5, l=3. This is calculated by total number of Victories/Number of Victories + losses for each generation. X axis is the script Id and Y axis is the average victories.



l=1



l=2



l=3

- 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 big 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 high counter value comparing to the other rule's counter value in same script. 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.

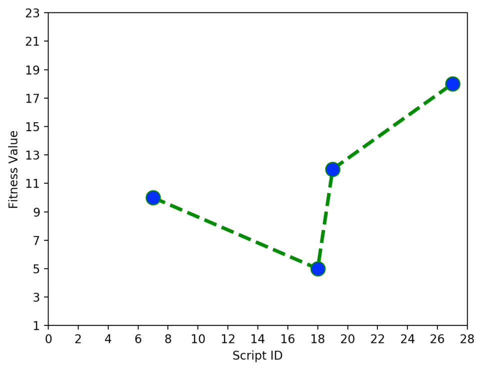
Experiment	Winning Script	Highest Counter value Rules	Lowest Counter value rules

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 Fitness. 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.

- 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.
    - how we can use different approaches in synthesizing script to use it for different other games.