How the Raccoons Colluded in the 2016 Rodent Election

Joshua Stallings

November 2020

1 Continuous Genetic Algorithm

With the string of votes given for Capybara, the squirrel votes were made with 100-[Capybara Votes]. The map of different precincts and the map of voters are kept separate, because the first part of the algo is trying to find the best continuous solution.

```
pop = getPopulation()
while pop[0].fitness() < N*N-4:
     pop = getBest(pop)
     print(pop[-1].fitness())
     pop = rePopulate(pop)
     mutatePopulation(pop)
```

N is the magic number that makes an NxN county map with N precincts. getPopulation() gets 25 county maps with counties randomly assigned to the precincts. getBest() sorts the solutions according to this fitness function.

```
g = self.grid
n = len(g)
v = 0
for i in range(len(g)):
    island, count = label(g==i)
    m = max((island==i+1).sum() for i in range(count))
    v += m/count

return v
```

The for-loop essentially gets the number of "features", which is wherever the grid array is equal to "i". The closer the number v is to N*N, the more fit the solution is. rePopulate() makes an empty list that is filled by randomly picked crossovers from two solutions.

```
x = np.random.randint(0, 2, self.grid.shape)
# ~ #c = a*x+b*(1-x)
output = CountyMap()
output.grid = self.grid*x+other.grid*(1-x)
output._fix()
return output
```

The cross() method takes the two solutions and makes a random grid with the same dimensions, but with random 1s and 0s. Wherever there is a 1, the precinct from solution A is kept, and wherever there is a 0 the precinct from B is kept. Then a fix() is done which just makes sure there is not too many of one precinct or too little of another.

Then the populations are mutated, which is done by taking the last 10 solutions in the pop list and randomly swapping the precinct value of two counties.

This is repeated until a solution has the fitness N*N-4, which is 96 with a 10x10 grid because 10*10-4 is 96. Any number could be used, such as 100, it just depends on how perfect a solution is needed in certain amounts of time.

2 Gerrymandering Genetic Algorithm

```
while pop[-1].fitness2() != (N*N)+N-3:
    pop = getBest2(pop)
    print(pop[-1].fitness2())
    pop = rePopulate(pop)
    mutatePopulation2(pop)
```

Once the population with the most continuous solutions is found, those solutions are passed onto here. (N*N)+N-3 is the amount of continuous precincts (N*N), and +N is how many precincts voting for a desired candidate. For a candidate to win, +N has to be at least 6. In total, this fitness value has to be at least 106 for a 10x10 county map.

getBest2() is the same as getBest(), except it uses fitness2() to rank the solutions. The fitness2() uses fitness() but adds a few extra steps, too.

```
def fitness2():
    v = self.fitness()
    margins = self.margin()
    votes = (sum(margins))
    return votes+v

def margin():
    n = len(self.grid)
    total = self.totalVotes()
    precinctMargins = []
    for i in range(n):
```

```
if total[i] >= 51*n:
precinctMargins.append(1)
else:
precinctMargins.append(0)
return precinctMargins
```

fitness2() uses margin(), which in turn gets the list returned by totalVotes(which is a list of the total number of votes per precinct). margin() then goes through the list of total votes, and if a precinct has a number greater than 51*n, it gets assigned a 1. 51*n is used because it is assumed each county has the same population of people, and so the amount to win would be at least 51*n.

Then, fitness2() adds up the number of 1s in the list from margin(). This is a way to represent which candidate won each precinct.

Other than that, each function in the while-loop is essentially the same. mutatePopulation2() is the same as mutatePopulation(), but copied so the number of solutions chosen to mutate can be customized for both while-loops.

3 Solutions

Some of the two solutions, one for capybara and one for squirrel, found were these.

Remember, for a precinct to vote for the desired candidate it needs to have 51*10, or 510 votes at least.

Capybara Solution:

8	8	8	3	3	3	1	5	5	5
8	8	8	3	3	3	1	6	5	5
2	8	8	3	3	1	1	6	5	5
2	8	8	3	3	1	1	6	6	5
2	2	2	0	0	0	1	6	6	5
2	4	2	2	0	0	1	6	6	5
7	4	2	2	4	0	1	6	6	9
7	4	4	4	4	0	1	9	9	9
7	4	4	4	7	0	0	0	9	9
7	7	7	7	7	7	9	9	9	9

The overall voter turnout was: $[450, 512, 522, 512, 522, 531, 522, 523, 512, 474]$

where each number represents the precincts like this: $[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]$ so precinct 1 had 512 votes, 0 had 450 votes, and so on. Overall, the Capybaras won by a margin of 8-2.

Squirrel Solution:

2	2	2	2	2	2	9	9	9	9
7	7	7	2	2	8	9	9	9	9
7	7	7	2	2	8	8	8	8	9
7	1	7	7	8	8	8	4	8	9
1	1	7	1	1	4	8	4	4	4
1	1	1	1	3	4	4	4	6	4
3	3	3	1	3	6	6	4	6	6
0	3	3	3	3	5	6	6	6	5
0	0	0	0	3	5	6	5	6	5
0	0	0	0	0	5	5	5	5	5

Overall Votes: $[511, 511, 469, 429, 511, 512, 512, 511, 513, 441]$

Unfortunately, the squirrel's margin was not as high. In this solution they had a margin of 7-3.