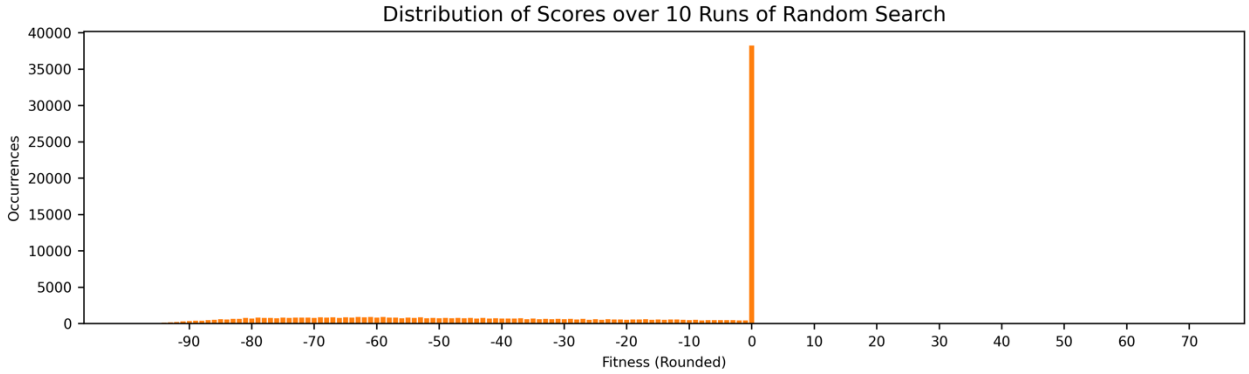
Mohab Yousef
COMP 5660 Fall 2024 Assignment 2a
Mey0012@auburn.edu

For the Green Part

Here are the configs that has been used:
I also sat C_Range between (-2, 2) in tree_genotype.py

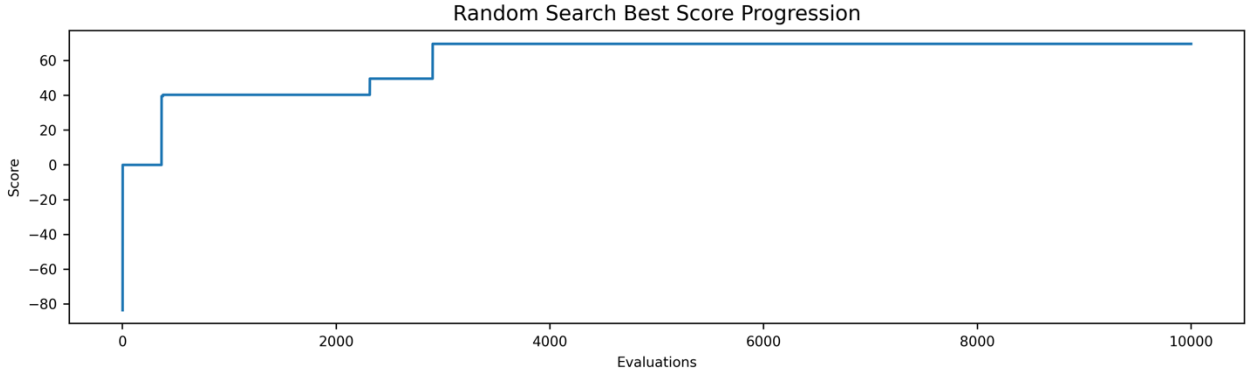
```
[pac_init]
depth_limit = 7

# Feel free to change this to a different representation,
# such as directly storing references to a function or class,
# like how we specified selection algorithms in series 1.
# Note you don't HAVE to use these variables, though we recommend it.
# You could instead hardcode your primitive set, if you wish.
terminals = ('G', 'P', 'F', 'W', 'C', 'M')
nonterminals = ('+', '-', '*', '/', 'RAND')


# Don't touch any of these
[game]
game_map = map.txt
pill_spawn = waves
pill_density = 1/3
fruit_prob = 1/75
fruit_score = 10
fruit_spawn = corners_spawned
time_multiplier = 2.5

```

Here is the histogram it displays the distribution of scores over 10 runs of random search



The histogram indicates that most of the controllers did not perform well because they have score close to 0, and only small numbers achieved high scores which mean that random search doesn't find strong solutions.

Eval v Game Score Start Step Plot



From the plot we can see that the evolutions started with negative scores and the best score increased in the first few thousand evaluations then it stopped and stayed the same, which mean that the random search found some found some good controllers ate the beginning but couldn't find better or improve after that.

Analysis:

Overall Best Score: 45.736434108527135
Mean of Best Scores: 18.22560354374308
Standard Deviation of Best Scores: 18.469382462671163

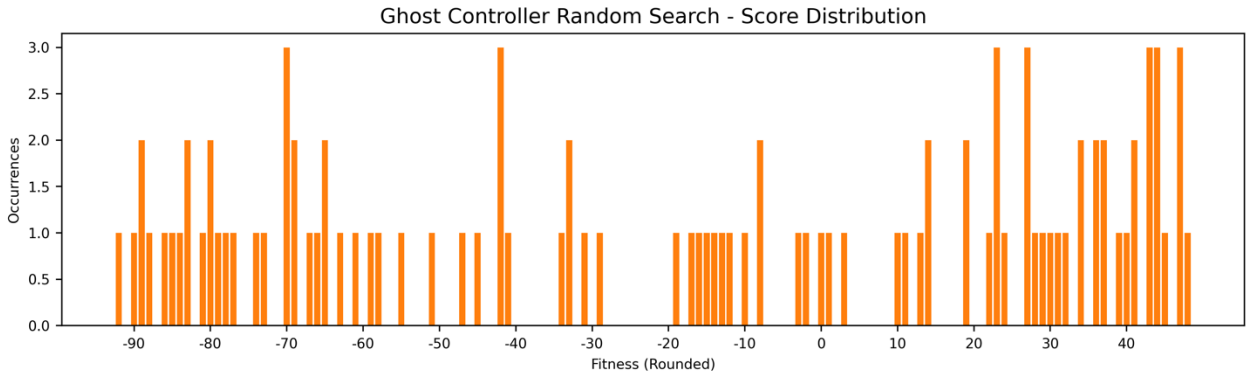
most runs didn't find very strong controllers, as the mean of best scores were much lower than the best score, std is 18.4 which is high which means random search gave very different results each time, sometime good solutions, and sometimes bad solutions.

For RED Part 1

Config Used

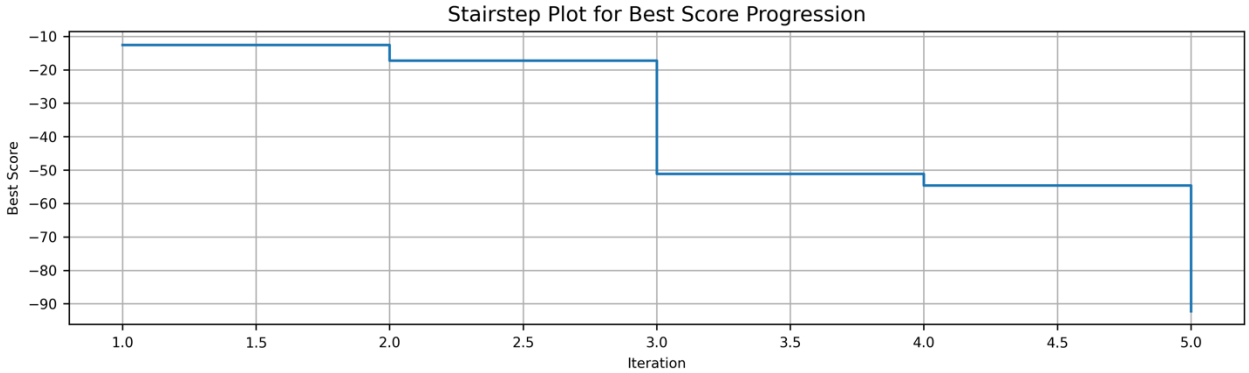
```
config['ghost_init'] = {
    'depth_limit': 7,
    'terminals': ('M', 'G', 'F', 'W', 'C'),
    'nonterminals': ('+', '-', '*', '/', 'RAND')
}
```

Histogram



The scores vary between -90 to 40 which means some ghost controllers were low and did well to capture the Pac-Man while others were high and did poorly.

The Stairstep Plot



As we see in the plot the scores started around -10 then when more ghost controllers were evaluated, the best score dropped, we can see that it too many evaluations for the ranmdo search to improve.

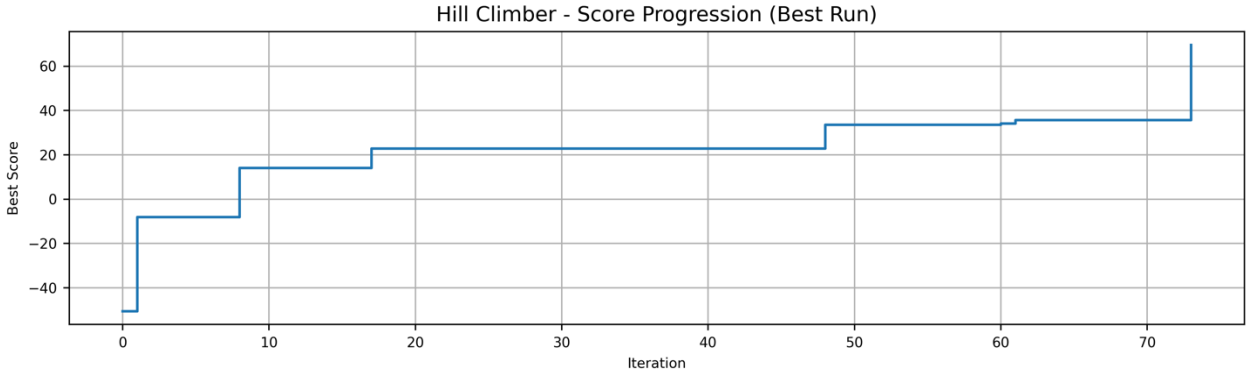
Analysis:
Overall Best Score: -94.12403100775194 (most effective to capture Pac-Man)
Mean Best Score: -91.85891472868217
Standard Deviation of Best Scores: 1.6428306563889605 (random search got similar results each time)

For RED Part 2 (the Hill Climber)

Configs Used

```
[pac_init]
depth_limit = 7
terminals = ('G', 'P', 'F', 'W', 'C')
nonterminals = ('+', '-', '*', '/', 'RAND')

[game]
game_map = map.txt
pill_spawn = waves
pill_density = 1/3
fruit_prob = 1/75
fruit_score = 10
fruit_spawn = corners_spawned
time_multiplier = 2.5

[hill_climber]
mutation_probability = 0.1
mutation_scale = 0.5
```

Stairstep Plot



The stairstep plot shows how the best score improves over iterations at higher iterations the improvements become smaller and slower which means the hill climber is has gotten closer to the best possible solution
Overall Best Score (Hill Climber): 69.6124031007752
Mean Best Score (Hill Climber): 6.961240310077519
Standard Deviation of Best Scores (Hill Climber): 20.883720930232556

Comparing hill climbing and random search:
Hill climbing found a higher top score (69.61) than random search (45.74), meaning hill climbing was able to find a better solution in one of the runs.
Random search had a higher average score (18.23) across runs compared to hill climbing average (6.96) meaning random search was more consistent

The author acknowledges the use of [ChatGPT -4] developed by [Open AI] in the preparation of this assignment. It was used in the following way(s): [code generation, erors fixing, and fixing grammar In the report]