Sujay Jakka

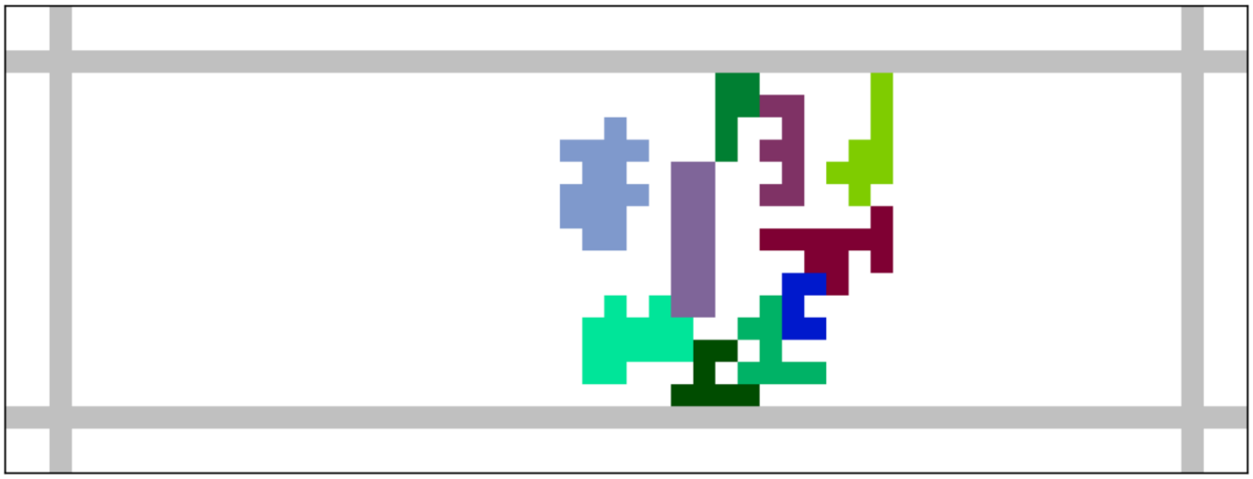
Svj0007@auburn.edu

COMP 5660 Fall 2024 Assignment 1b

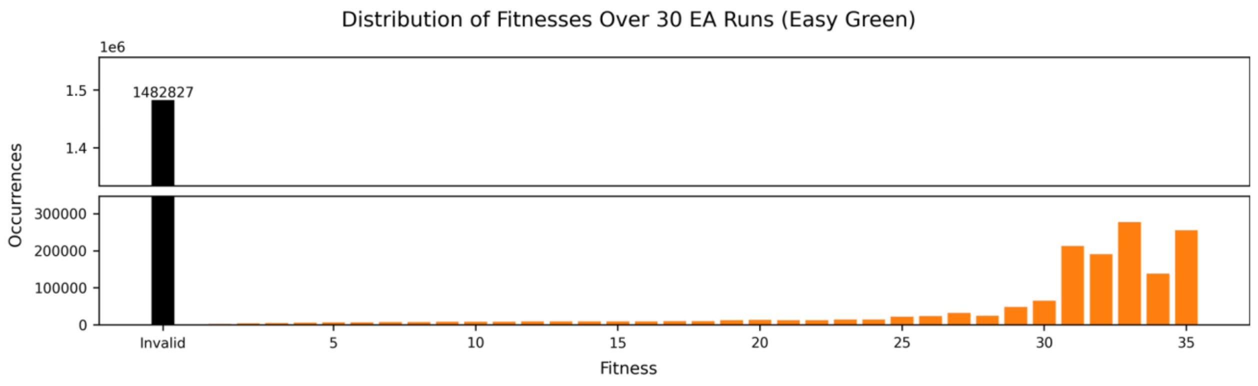
22 September 2024

Assignment 1b Report

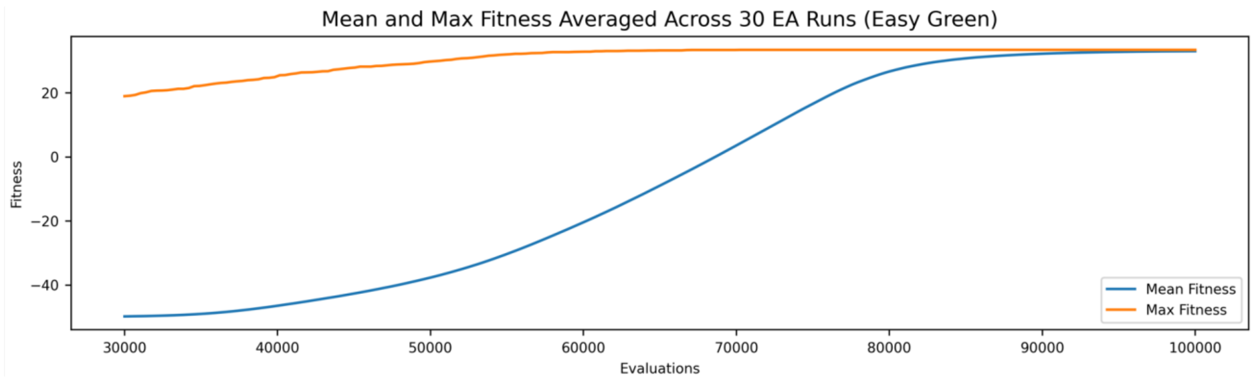
I will first introduce the results for the easy green problem instance. The best fitness my EA achieved across 30 runs was a fitness of 35. This solution yielded a higher fitness than the best fitness of my random search algorithm in assignment 1a, where I had a best fitness of 25. The image below shows the visualization of the best solution from my EA.



Looking at the distribution of fitness over 30 runs of my EA, I had 1482827 invalid solutions which was approximately 49.43% of all solutions. The proportion of invalid solutions was a lot lower than the random search algorithm where 99.63% of solutions were invalid. The image below is the histogram showing the distribution of fitness values across 30 runs.



The image below describes the mean and max fitness of the EA averaged across 30 runs. From the plot below, it can be seen that the mean fitness curve converged with the max fitness curve at 95000 evaluations which means the EA did not exhibit premature convergence.

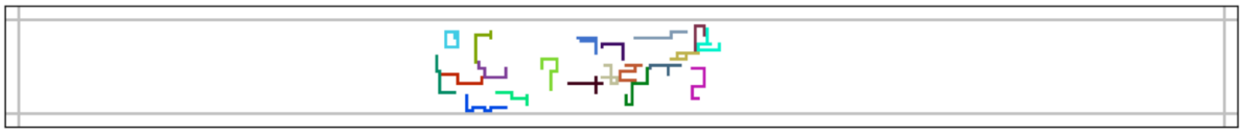


Furthermore, the mean of the sample of 30 runs(sample size 30) of the EA was 33.47, while the mean of the sample of 30 runs(sample size 30) of the random search algorithm was 21.1. The standard deviation of the EA and random search sample was 1.48 and 1.75 respectively. I performed a two sample independent t-test where the null hypothesis was that the two algorithms are equally effective at solving the problem, and the alternative hypothesis is that the two algorithms are not equally effective at solving the easy green problem instance. I performed the t-test with a significance level(alpha) of 0.05 and a degrees of freedom of 30. I got a p-value of 4.81309e-36, which is significantly smaller than the significance level of 0.05. We can then reject the null hypothesis, and claim that the two algorithms have statistically significant differences in performance. Because it is highly unlikely that both sample distributions have the same population mean and that the EA sample has a higher mean than the random search sample, we can conclude that the EA performed better. The parameters I used for the easy green problem instance are shown below.

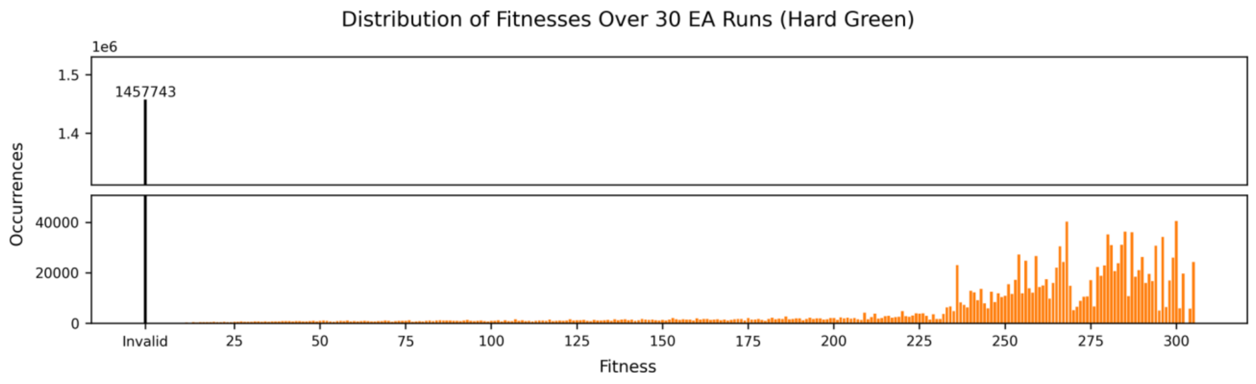
```
--------------------------------------------------------------------------------------------------------------
[ea]
# mu value for around 100 valid solutions, old value 99_000
mu = 30_000
# increasing children from 50 to 350
num_children = 350
# mutation_rate .01, 1000 times more likely than the previous value .00001
mutation_rate = 0.01
parent_selection = k_tournament_with_replacement
survival_selection = k_tournament_without_replacement

[recombination_kwargs]
# One point crossover, preserve positional bias, old method uniform
method = one-point
prob_selecting_parent_1_gene = 0.5

[parent_selection_kwargs]
# Increase k from 1 to 50, increasing parent selection pressure
k = 50
```

```
[survival_selection_kwargs]

# Increase k from 1 to 75, increasing survival pressure

k = 75


[mutation_kwargs]

# Increasing the probability of random reset of the rotation from .1 to .2

prob_random_reset = .2

mean_of_change_dist = 0

# Changing std from 1 to 2, increasing probability of larger change

std_of_change_dist = 2
```

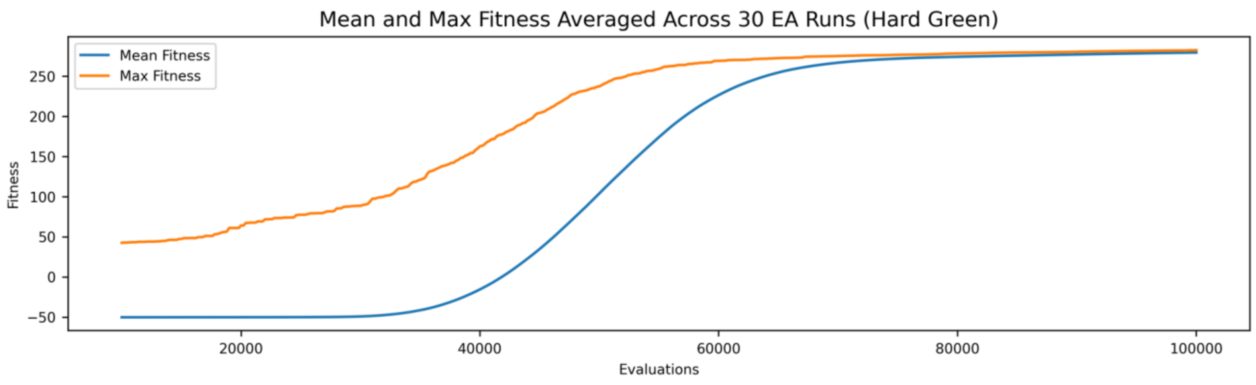
Moving on to the hard green problem instance, the best fitness my EA achieved across 30 runs was a fitness of 305. This solution yielded a higher fitness than the best fitness of my random search algorithm in assignment 1a, where I had a best fitness of 123. The image below shows the visualization of the best solution from my EA.



Looking at the distribution of fitness over 30 runs of my EA, I had 1457743 invalid solutions which was approximately 48.59% of all solutions. The proportion of invalid solutions was a lot lower than the random search algorithm where 99.97% of solutions were invalid. The image below is the histogram showing the distribution of fitness values across 30 runs.



The image below describes the mean and max fitness of the EA averaged across 30 runs. From the plot below, it can be seen that the mean fitness curve converged with the max fitness curve at 85000 evaluations which means the EA did not exhibit premature convergence and it was able to explore the solution space a decent amount.



Furthermore, the mean of the sample of 30 runs(sample size 30) of the EA was 282.8, while the mean of the sample of 30 runs(sample size 30) of the random search algorithm was 89.3. The standard deviation of the EA and random search sample was 16.70 and 20.52 respectively. I performed a two sample independent t-test similar to the easy green problem, where the null hypothesis was that the two algorithms are equally effective at solving the problem, and the alternative hypothesis is that the two algorithms are not equally effective at solving the hard green problem instance. I performed the t-test with a significance level(alpha) of 0.05 and a degrees of freedom of 30. I got a p-value of 9.40618e-43, which is significantly smaller than the significance level of 0.05. We can then reject the null hypothesis, and claim that the two algorithms have statistically significant differences in performance. Because it is highly unlikely that both sample distributions have the same population mean and that the EA sample has a higher mean than the random search sample, we can conclude that the EA performed better. The parameters I used for the hard green problem instance are shown below.

```
----------------------------------------------------------------------------------------------------------------
[ea]
# mu value for around 2 valid solutions, old value 99_000
mu = 10_000
# increasing children from 50 to 200
num_children = 200
# mutation_rate .1, 10000 times more likely than the previous value .00001
mutation_rate = 0.1
parent_selection = k_tournament_with_replacement
survival_selection = truncation

[recombination_kwargs]
method = uniform
prob_selecting_parent_1_gene = 0.5

[parent_selection_kwargs]
# Increase k from 1 to 30, increasing parent selection pressure
k = 30
```

```
[survival_selection_kwargs]
k = 1

[mutation_kwargs]
# Increasing the probability of random reset of the rotation from .1 to .2
prob_random_reset = .2
mean_of_change_dist = 0
std_of_change_dist = 1
```