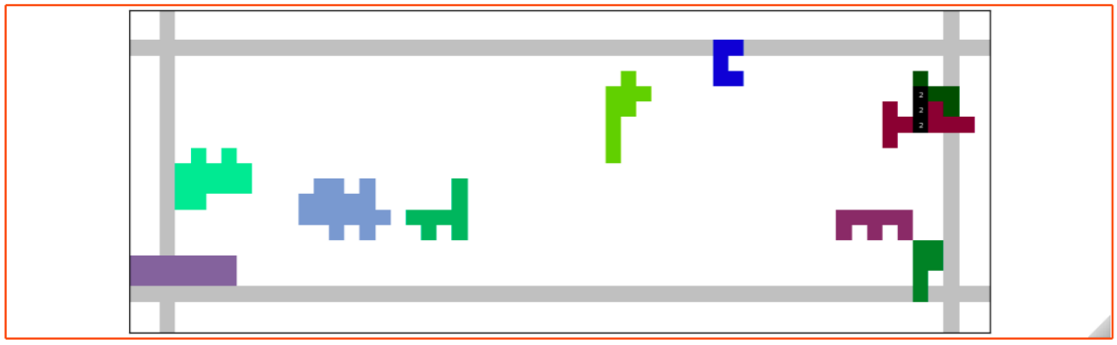
Assignment 1B report

Name: Mohab Yousef
Mey0012

Test Linear Genotype

```
print(f'The solution has fitness {test_solution.fitness}, and looks like this:')
visualize(test_solution.genes, **config['problem'])

del config, test_solution, output
```

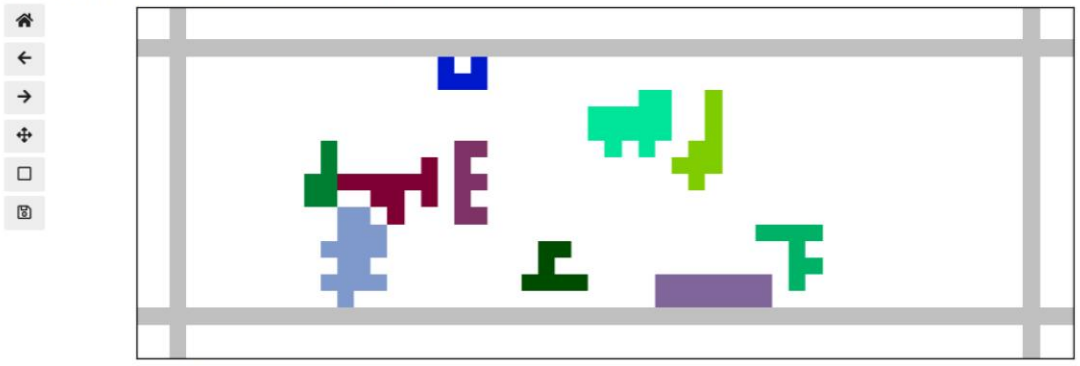
```
The random_initialization function did something? True
Was the solution the right length? True
The solution has fitness -50, and looks like this:
```



Fitness Evolution

```
    print('The fitnesses were distributed like this:')
    hist.get_plot('Example Fitness Distribution').show()

    del config, hist, example_population, fitnesses, best_individual
```

```
Mean fitness of population: -49.7942
Best fitness in population: 19
The highest-fitness individual looks like this:
```



```
The fitnesses were distributed like this:
```

Example Fitness Distribution

Fitness Distribution.

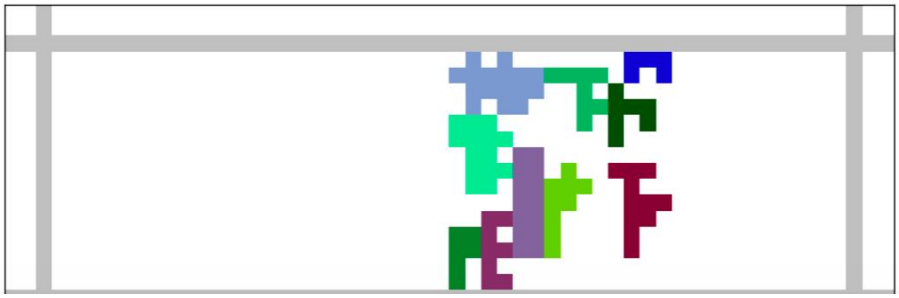
Aside: Sampling With and Without Replacement

Stochastically selecting samples from a population is a very common practice in many fields of study, and a critical component of EC implementations. These

Calling EAs log_base_state and adding fitness to Histogram

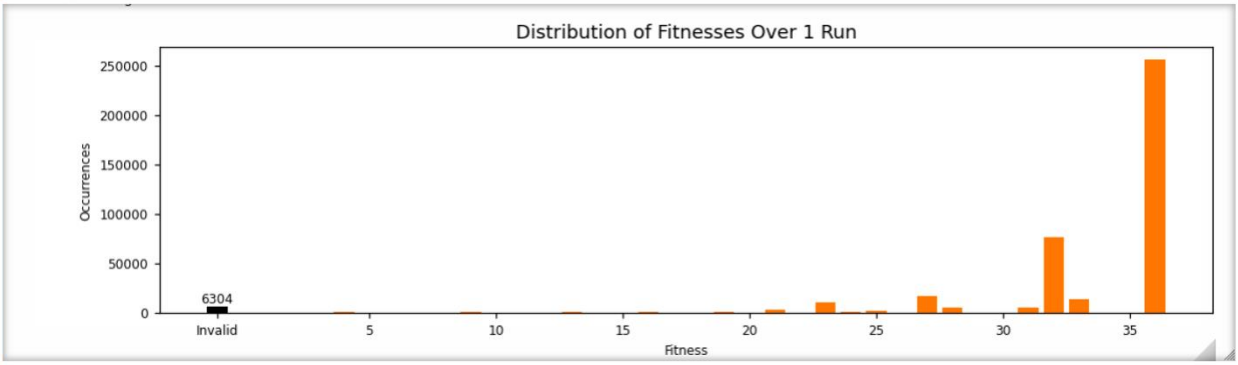
```
print('Max fitnesses per generation:')
print(max_fit_per_gen)
print('Evaluation counts per generation:')
print(evaluation_counts)

del config, num_evaluations, hist, log, best_solution, mean_fit_per_gen, max_fit_per_gen, evaluation_counts
```

```
Best solution fitness: 36
Best solution looks like:
```

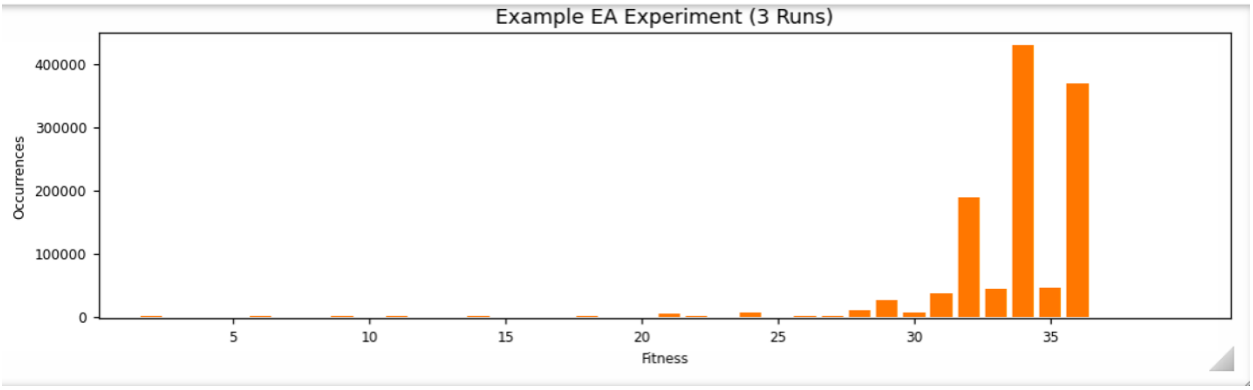


Distribution fitness over 1 run Histogram

```
print('Max fitnesses per generation:')
print(max_fit_per_gen)
print('Evaluation counts per generation:')
print(evaluation_counts)

del config, num_evaluations, hist, log, best_solution, mean_fit_per_gen, max_fit_per_gen, evaluation_counts
```



Example EA Experiment for 3 runs



Easy Green Config

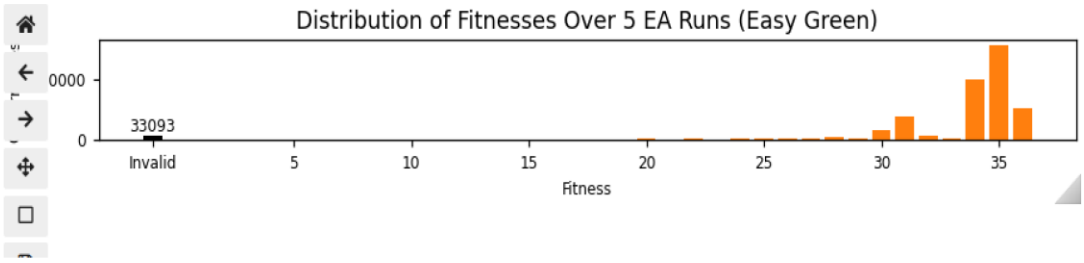
jupyter easy_green_config.txt Last Checkpoint: 3 days ago

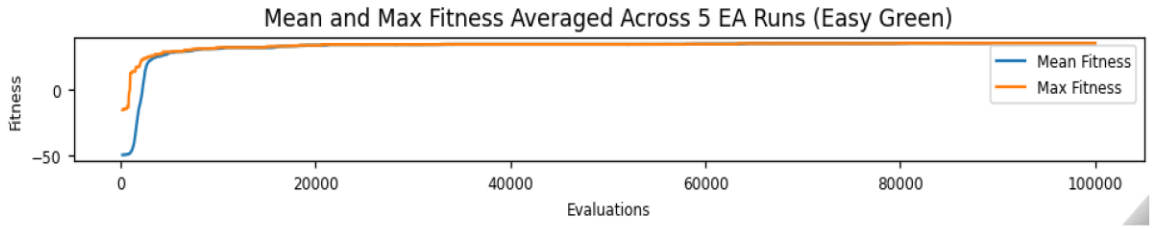
File Edit View Settings Help

```
[ea]
mu = 200
num_children = 50
mutation_rate = 0.05
parent_selection = k_tournament_with_replacement
survival_selection = k_tournament_without_replacement
# Don't touch this
individual_class = LinearGenotype

[recombination_kwargs]
method = uniform

[parent_selection_kwargs]
k = 3

[survival_selection_kwargs]
k = 2
```

Easy Green Average best fitness = 35.2 when mu = 200, num children =50, mutation rate = 0.05 over 5 runs

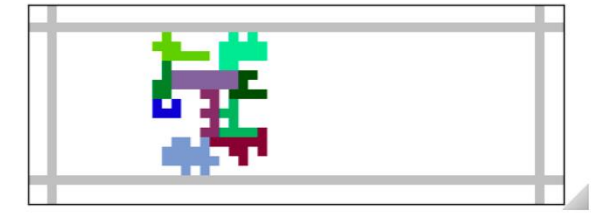
```
Data saved to tuning/1b/easy_green_config/3
```

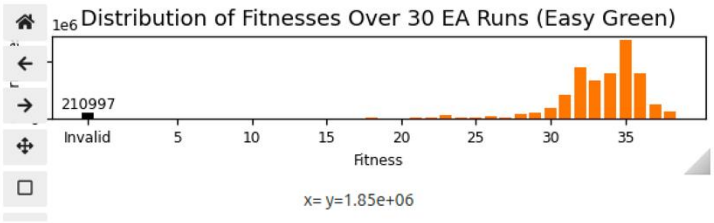


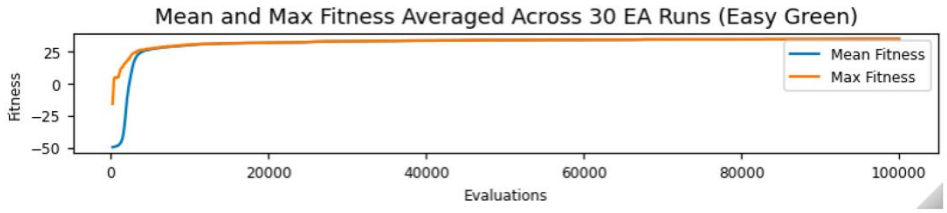



```
Average best fitness: 35.2
```

EA for easy green over 30 EA runs







Statical Analysis for Easy Green

```
[34]: run_stats('data/1a/easy/best_per_run.txt', 'data/1b/easy_green/best_per_run.txt')

Number of samples: 30
data/1a/easy/best_per_run.txt mean: 20.433333333333334
data/1a/easy/best_per_run.txt stdv: 2.0625283000496615
data/1b/easy_green/best_per_run.txt mean: 35.03333333333333
data/1b/easy_green/best_per_run.txt stdv: 1.6914252984479938
p-value: 4.066667372754878e-36
```

Hard Problem Instance

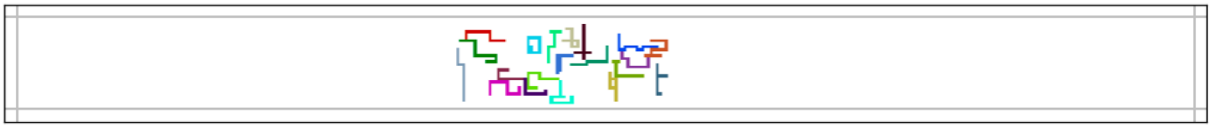
1a Average = 20.43, Variation = 2.06
1b: Average = 35.03, Variation = 1.69
1b data performs much better than 1a

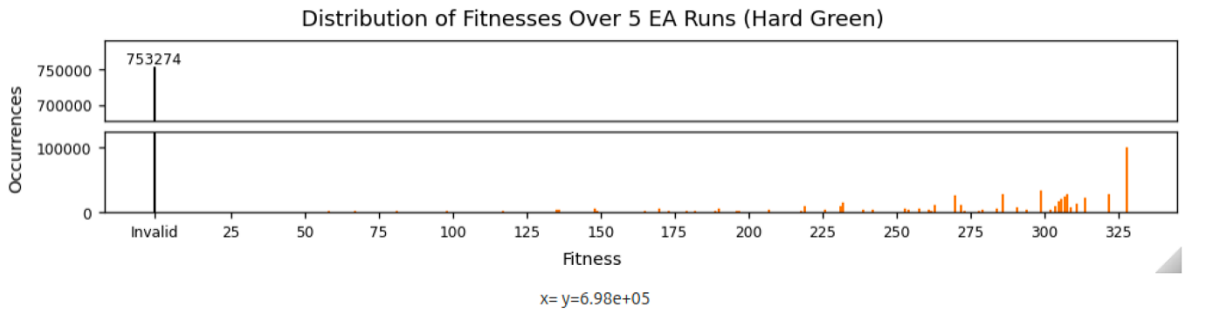
EA over 5 EA Hard Green, mu =50, number of children = 20, mutation rate = 0.05

jupyter hard_green_config.txt Last Checkpoint: 3 days ago

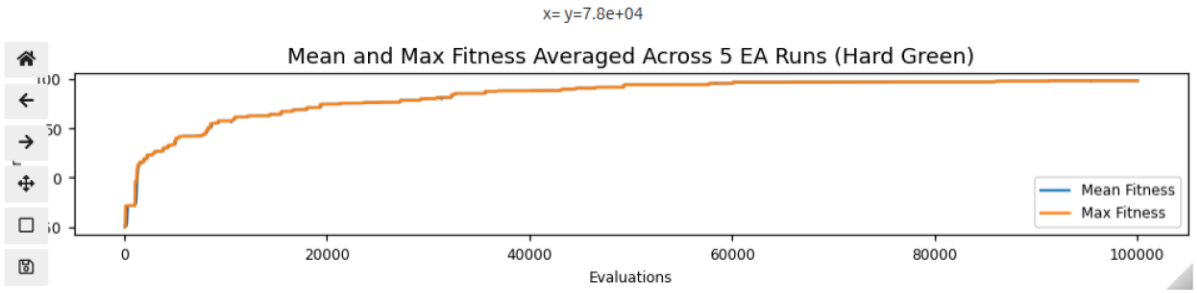
File Edit View Settings Help

```
[ea]
mu = 50
num_children = 20
mutation_rate = 0.05
parent_selection = k_tournament_with_replacement
survival_selection = k_tournament_without_replacement
# Don't touch this
individual_class = LinearGenotype

[recombination_kwargs]
method = uniform

[parent_selection_kwargs]
k = 4

[survival_selection_kwargs]
k = 3

[mutation_kwargs]
bonus = False
# Don't touch this
bounds = ${problem:bounds}
```



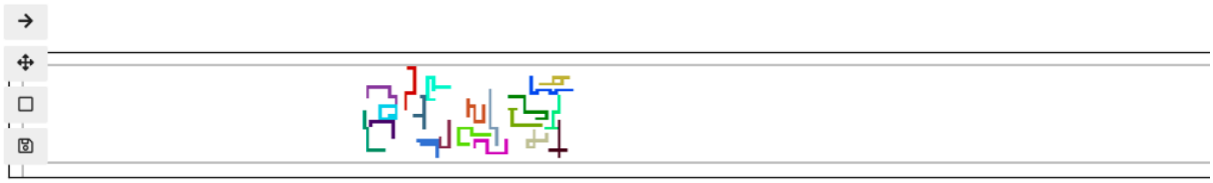


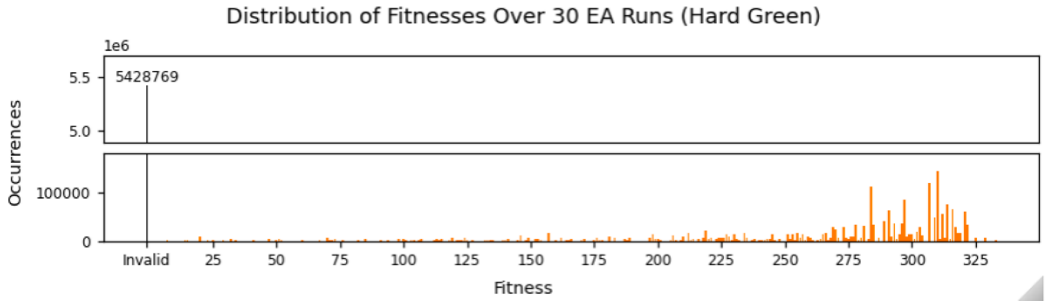
Average Best fitness = 98.4



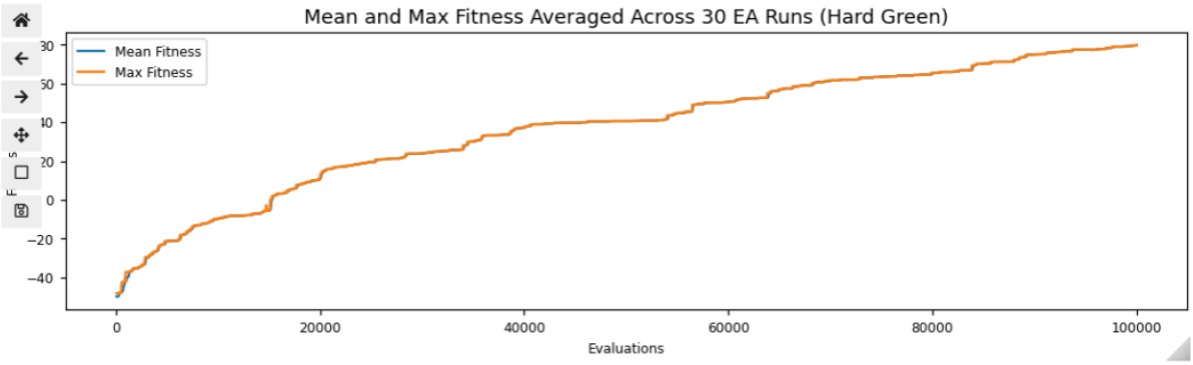

```
Average best fitness: 98.4
```

EA over 30 runs for Hard Green





Evolution and Statical analysis are down below shows the mean, stdv, best run mean, best run stdv, and p-value




```
Number of samples: 30
data/1a/hard/best_per_run.txt mean: 84.36666666666666
data/1a/hard/best_per_run.txt stdv: 14.772956030702098
data/1b/hard_green/best_per_run.txt mean: 81.53333333333333
data/1b/hard_green/best_per_run.txt stdv: 173.44595368752397
p-value: 0.9295663083714747
```

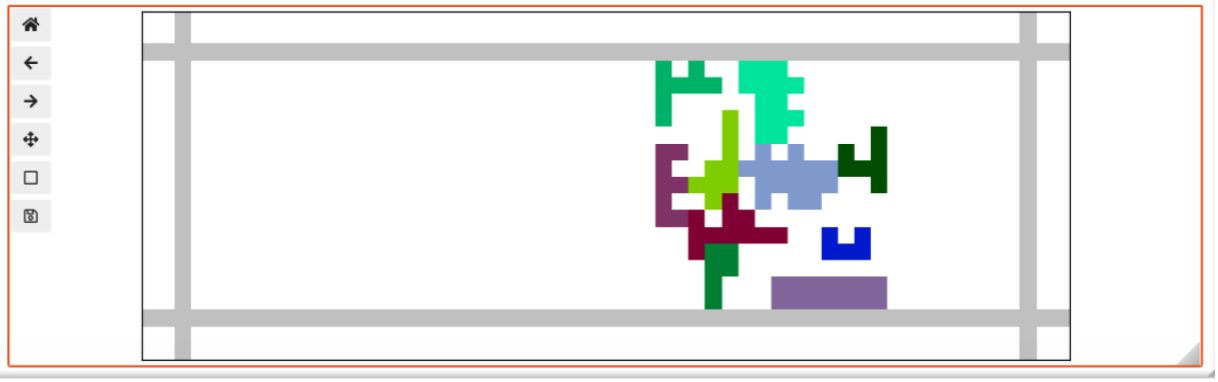
1a (hard):
Average = 84.37
Variation = 14.77
1b (hard green):
Average = 81.53
Variation = 173.45 (much higher variation)
The p-value is 0.93, which is quite large. This means there is no significant difference between the two datasets.

Easy Yellow Best solution fitness is 36

```
[ea]
mu = 30
num_children = 10
mutation_rate = 0.05
parent_selection = stochastic_universal_sampling
survival_selection = k_tournament_without_replacement
# Don't touch this
individual_class = LinearGenotype

[recombination_kwargs]
method = uniform

[parent_selection_kwargs]
k = 3

[survival_selection_kwargs]
k = 2

[mutation_kwargs]
bonus = False
# Don't touch this
bounds = ${problem:bounds}
```

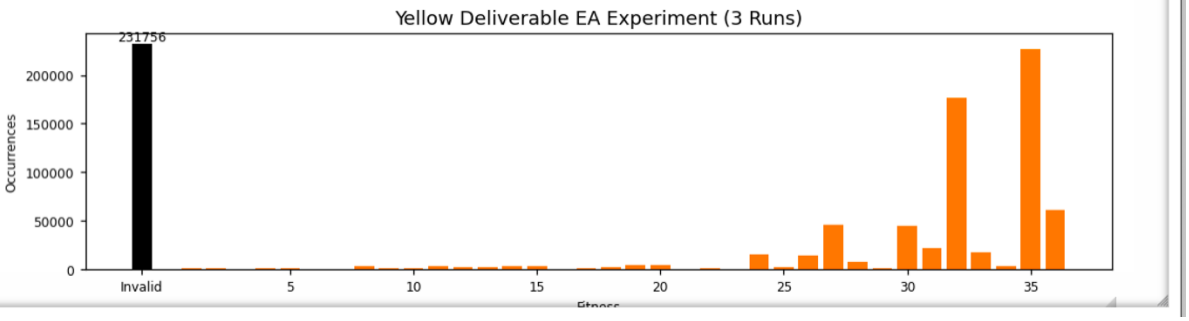
```
print('Evaluation counts per generation:', yellow_evaluation_counts)
```

```
Mean population fitness dropped significantly over time at least once. This *may* indicate a bug (especially if using truncatio
n), or poor configuration. You may ignore this if you deliberately chose a very non-elitist configuration and can justify your ch
oice.
Best fitness per run: [35, 36, 32]
Best solution fitness: 36
```



Easy Yellow Statistical Analysis

```
Statistical Analysis of Best Fitnesses (Yellow Deliverable):
Mean Best Fitness: 34.333333333333336
Standard Deviation of Best Fitness: 2.0816659994661326
Maximum Best Fitness: 36
Minimum Best Fitness: 32
```



Mean Best Fitness: 34.33
Standard Deviation: 2.08 (shows the results are fairly close to the average)
Maximum Best Fitness: 36
Minimum Best Fitness: 32
the fitness scores are quite consistent, staying within a narrow range (32 to 36).

Hard Yellow Config

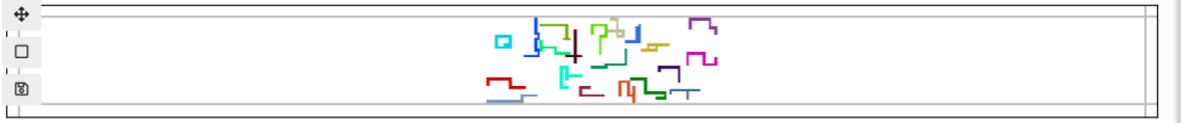
jupyter hard_yellow_config.txt Last Checkpoint: 3 days ago

File Edit View Settings Help

```
[ea]
mu = 100
num_children = 35
mutation_rate = 0.05
parent_selection = stochastic_universal_sampling
survival_selection = k_tournament_without_replacement
# Don't touch this
individual_class = LinearGenotype

[recombination_kwargs]
method = uniform

[parent_selection_kwargs]
k = 5

[survival_selection_kwargs]
k = 4

[mutation_kwargs]
bonus = False
# Don't touch this
bounds = ${problem:bounds}
```

Best solution Fitness is 318 for Hard Yellow

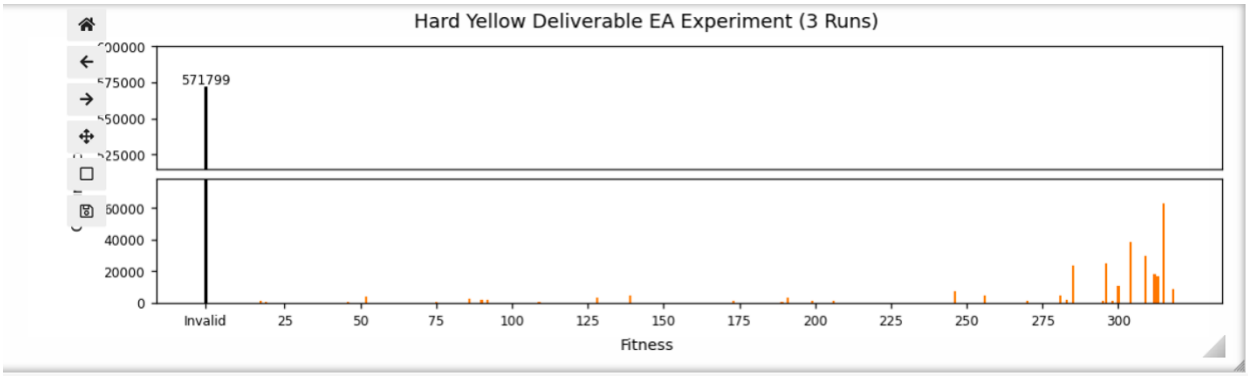
```
Best fitness per run (Hard Yellow): [318, -50, -50]
Best solution fitness (Hard Yellow): 318
```



Statical Analysis for Hard Yellow

Mean Best Fitness: 72.67
Standard Deviation: 212.46
Maximum Best Fitness: 318
Minimum Best Fitness: -50
some runs performed very poorly. high variation means that the performance is inconsistent.

```
Statistical Analysis of Best Fitnesses (Hard Yellow Deliverable):
Mean Best Fitness: 72.66666666666667
Standard Deviation of Best Fitness: 212.4648990617823
Maximum Best Fitness: 318
Minimum Best Fitness: -50
```

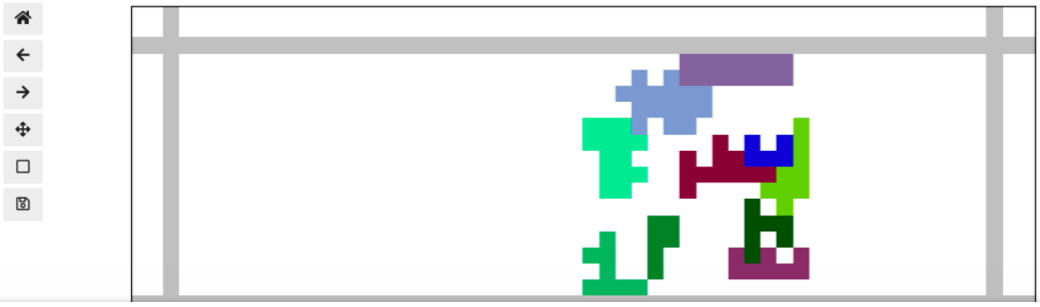


Easy Red Config

```
[ea]
mu = 40
num_children = 15
mutation_rate = 0.05
parent_selection = k_tournament_with_replacement
survival_selection = k_tournament_without_replacement
# Don't touch this
individual_class = LinearGenotype

[recombination_kwargs]
method = uniform

[parent_selection_kwargs]
k = 4

[survival_selection_kwargs]
k = 3

[mutation_kwargs]
bonus = False
# Don't touch this
bounds = ${problem:bounds}
```

Best Solution Fitness for Easy RED = 36

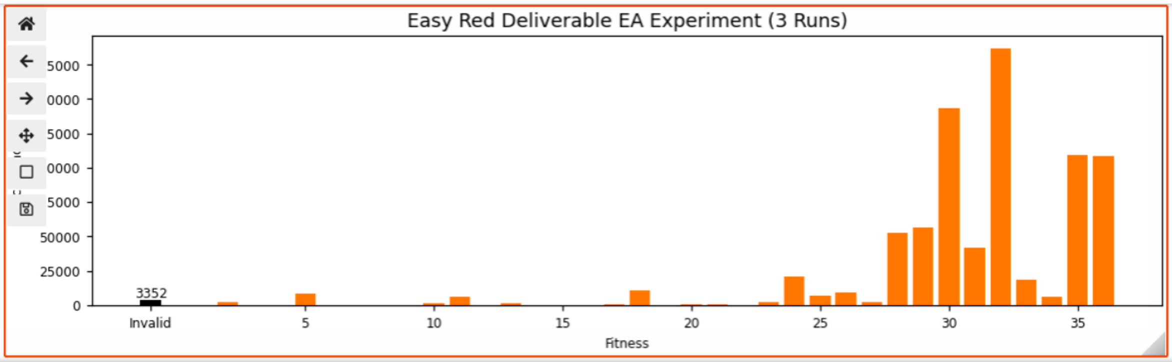
```
Best fitness per run (Easy Red): [36, 34, 32]
Best solution fitness (Easy Red): 36
```



Statical Analysis for Easy RED

```
Statistical Analysis of Best Fitnesses (Easy Red Deliverable):
Mean Best Fitness: 34
Standard Deviation of Best Fitness: 2.0
Maximum Best Fitness: 36
Minimum Best Fitness: 32
```

Mean Best Fitness: 34
Standard Deviation: 2.0 – Shows that the scores are close to the average
Maximum Best Fitness: 36
Minimum Best Fitness: 32
The fitness scores are quite consistent, staying within a narrow range of 32 to 36.



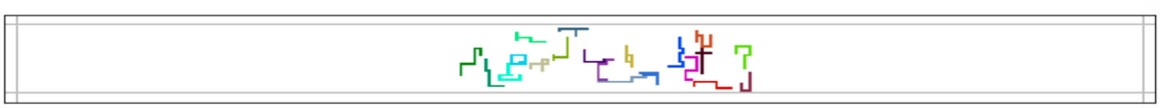

```
Mean fitnesses per generation per run (Easy Red): [[-50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -5
0, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -48.7, -48.7, -48.7, -48.7, -47.4, -44.8, -40.9, -35.7, -24,
-9.7, -0.6, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2.075, 2.075, 2.225, 2.45, 2.9, 3.425, 4.325, 4.925, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5
, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5
, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5.15, 5.45, 6.2, 7.425, 9.525,
10.85, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11
```

Hard Red Config

```
[ea]
mu = 70
num_children = 30
mutation_rate = 0.05
parent_selection = k_tournament_with_replacement
survival_selection = k_tournament_without_replacement
# Don't touch this
individual_class = LinearGenotype

[recombination_kwargs]
method = uniform

[parent_selection_kwargs]
k = 5

[survival_selection_kwargs]
k = 4
```

Hard Red Solution Fitness is 296

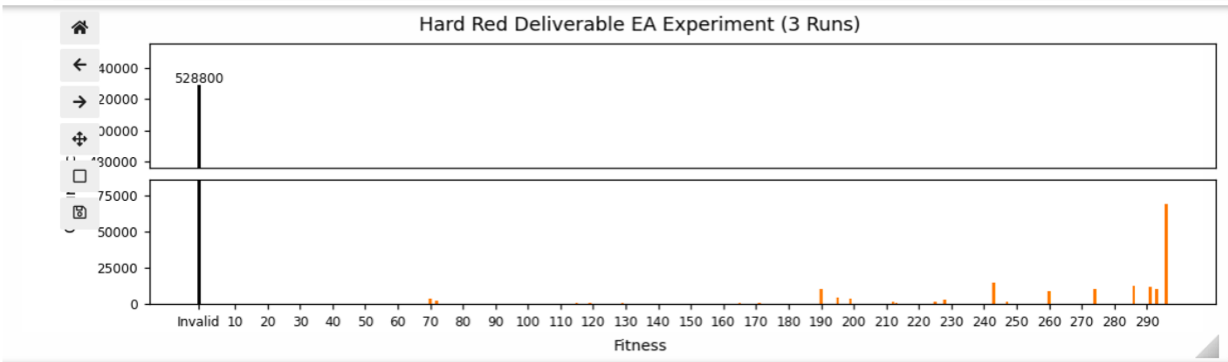
```
Best fitness per run (Hard Red): [-50, 296, -50]
Best solution fitness (Hard Red): 296
```



Statical Analysis For Hard RED

```
Statistical Analysis of Best Fitnesses (Hard Red Deliverable):
Mean Best Fitness: 65.33333333333333
Standard Deviation of Best Fitness: 199.7631931396105
Maximum Best Fitness: 296
Minimum Best Fitness: -50
```

Mean Best Fitness: 65.33
Standard Deviation: 199.76 very high variation
Maximum Best Fitness: 296
Minimum Best Fitness: -50
with both high and very low scores, we can see that it is inconsistent performance.




```
Mean fitnesses per generation per run (Hard Red): [[-50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -5
0, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50,
-50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50,
-50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50,
-50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50,
-50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50,
-50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50,
-50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50,
-50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50,
-50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50,
-50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50,
-50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50,
-50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50, -50,
```