COMP 5660
Mohab Yousef
904154154
Mey0012
Assignment 1C Report

Green_Config Paramter.

jupyter green_config.txt Last Checkpoint: 3 days ago

File Edit View Settings Help

```
[ea]
mu = 500
num_children = 100
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 = 2

[survival_selection_kwargs]
k = 3

[fitness_kwargs]
penalty_coefficient = 1/128
red = False

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

Unconstrianed fitness function

Edit View Run Kernel Settings Help

Markdown JupyterLab Python 3 (ipykern

```python
print('Violations:', test_solution.violations)
print('Penalized fitness:', test_solution.fitness)
print('Visualization:')
visualize(test_solution.genes, **config['problem'])

del config, penalty_coefficient, test_solution, output
```

```
Base fitness: -50
Violations: 33
Penalized fitness: 7.7421875
Visualization:
```

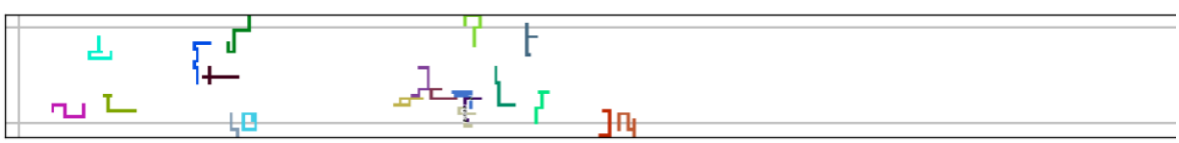
Note how we've assigned three member variables:

Edit View Run Kernel Settings Help

Markdown JupyterLab Python 3 (ipykern

```python
    print('Individuals with unassigned base fitness:', no_base)
    print('Individuals with unassigned violations:', no_violations)
    print('Individuals with unassigned fitness:', no_penalized)

del config, example_population, no_base, no_violations, no_penalized, failed
```

```
Average penalized fitness of population: 28.969018125
Best penalized fitness in population: 209.34375
Average base fitness of population: -49.97451
Best base fitness in population: 88
Number of valid solutions: 31
Visualizing solution with the best penalized fitness:
```



Visualizing solution with the best base fitness:

Edit View Run Kernel Settings Help

Code JupyterLab Python 3 (ipykern

```
    print('Individuals with unassigned base fitness:', no_base)
    print('Individuals with unassigned violations:', no_violations)
    print('Individuals with unassigned fitness:', no_penalized)

del config, example_population, no_base, no_violations, no_penalized, failed
```

```
Visualizing solution with the best base fitness:
```

Assembling your EA

Now you get to use the framework you implemented in Assignment 1b to build a constraint satisfaction EA! Note that this can be nearly identical to the function from Assignment 1b's notebook, with a couple key differences. First, make sure you're calling

EA

Edit View Run Kernel Settings Help

Code JupyterLab Python 3 (ipykern

```
print('Violation histogram:')
violation_hist.get_plot('Distribution of Violations Over 1 Run').show()

del config, num_evaluations, best_solution, mean_fit_per_gen, \
    max_fit_per_gen, mean_base_per_gen, max_base_per_gen, valid_per_gen, \
    evaluation_counts, hist, base_hist, violation_hist, log
```

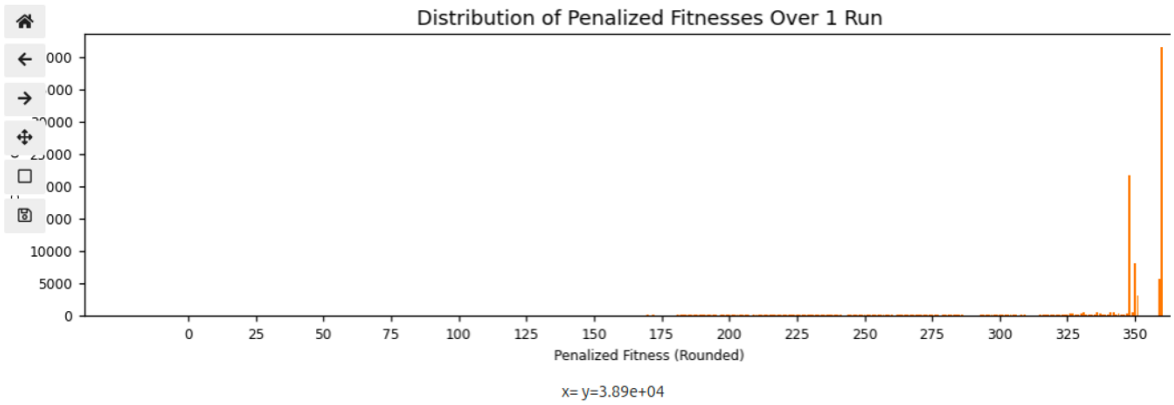
```
Best solution fitness: 83.0
Best solution base fitness: 83
Best solution looks like:
```

```
Penalized fitness histogram:
```

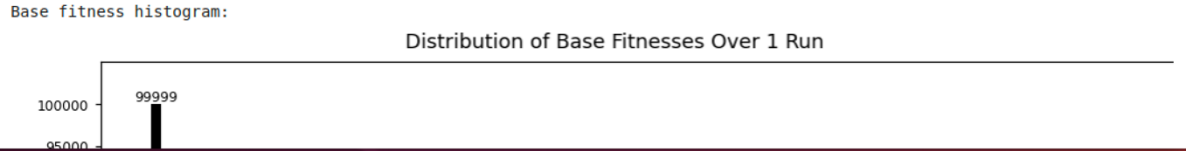
Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
Penalized fitness histogram:
```




```
Base fitness histogram:
```

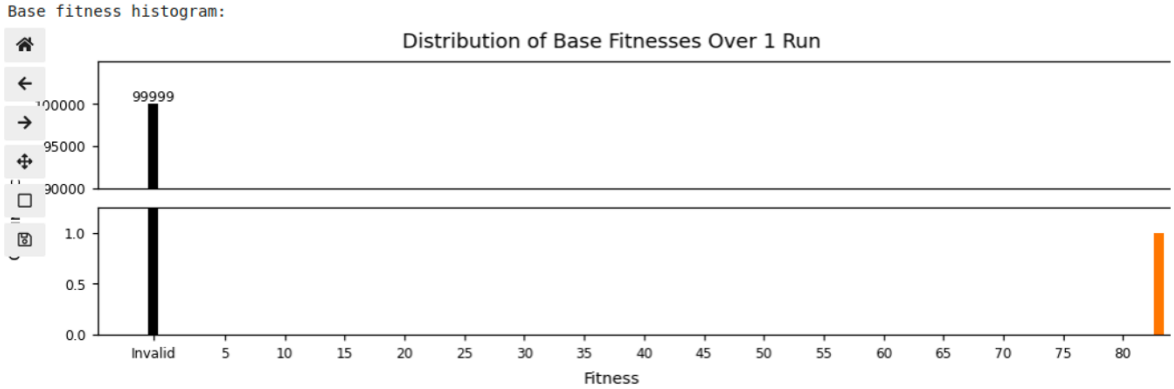


Edit View Run Kernel Settings Help Trusted

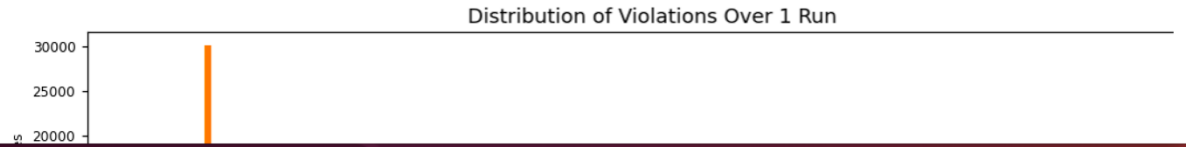
Code JupyterLab Python 3 (ipykernel)

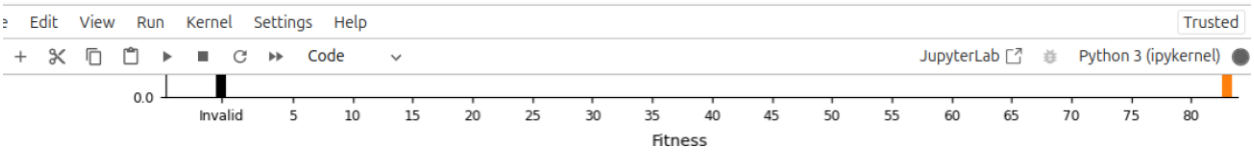
x= y=3.89e+04

Base fitness histogram:

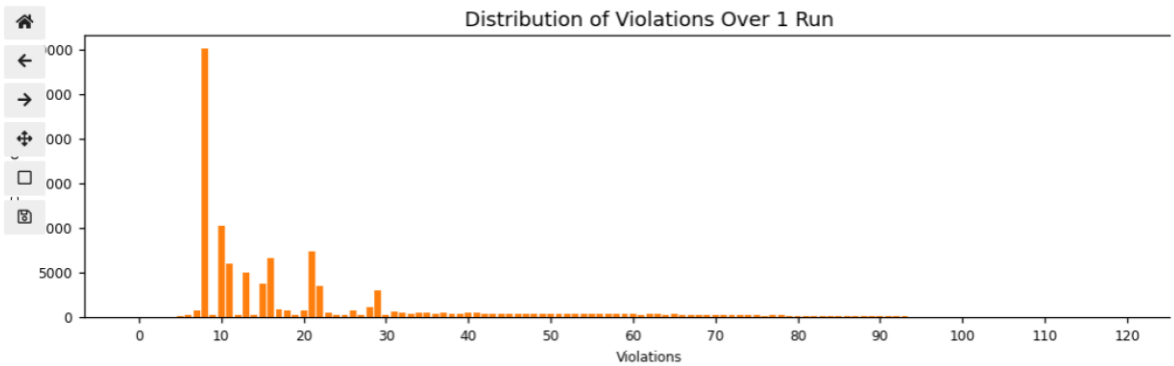



```
Violation histogram:
```



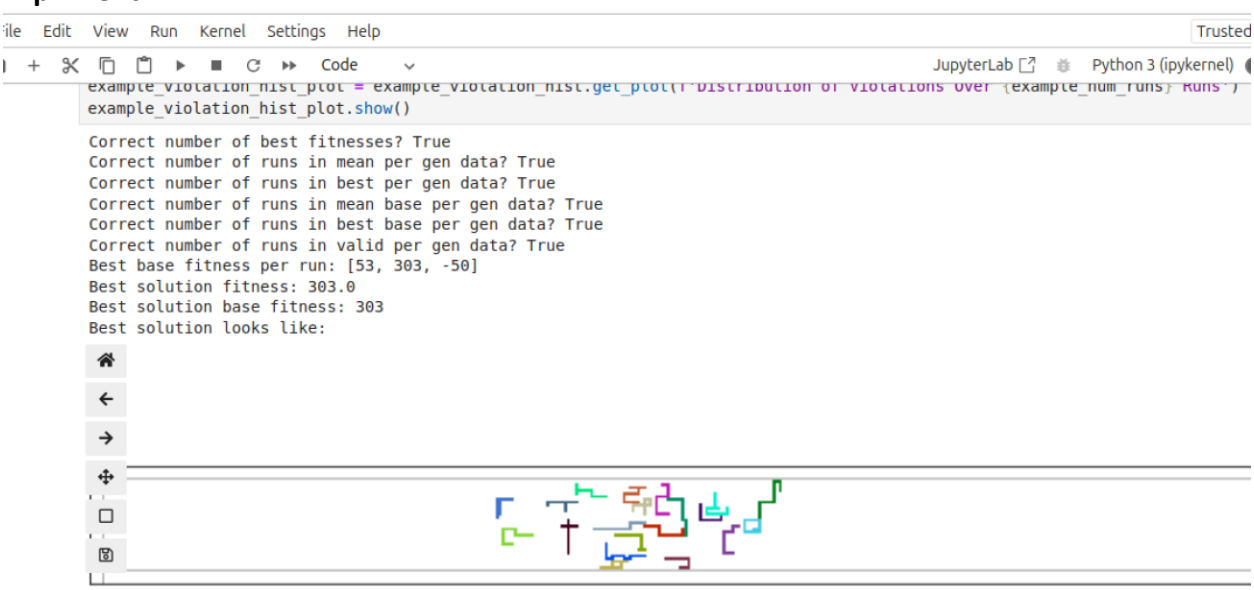


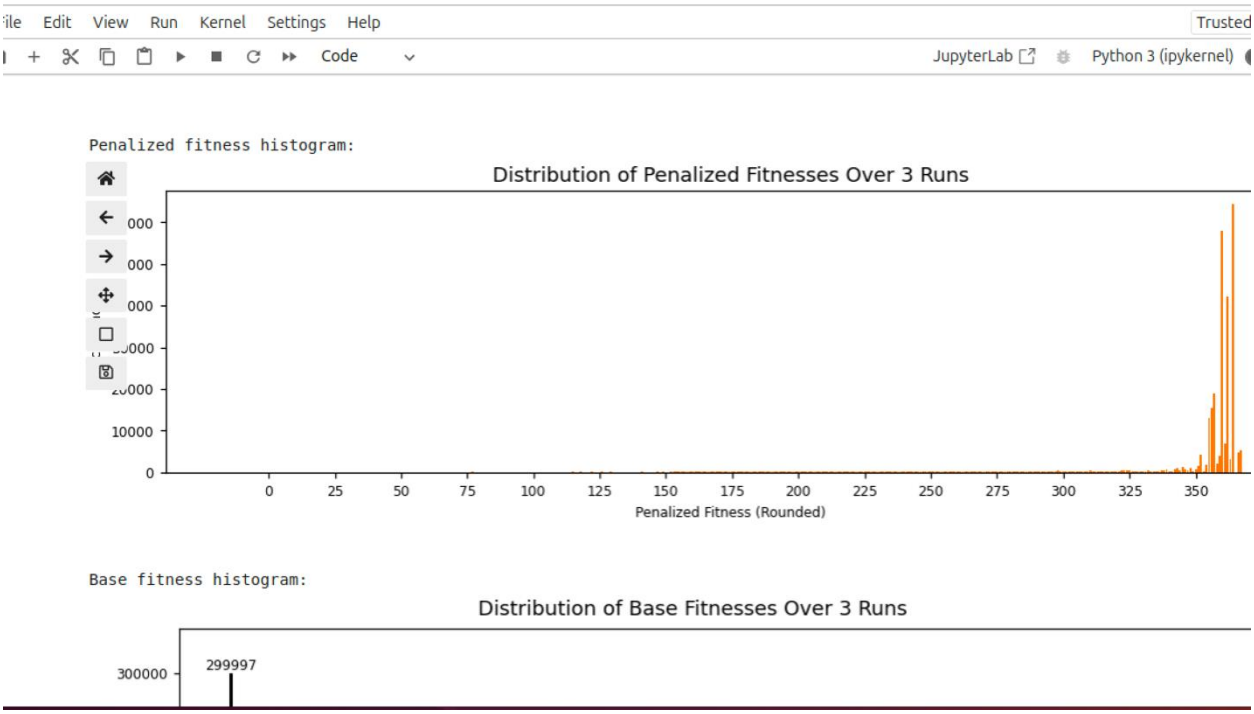

```
Violation histogram:
```



If your results are dissapointingly bad, don't worry; that's expected. You'll need to tune your EA later in order to obtain decent results, and we will walk you through this process later in this notebook.

Expriment

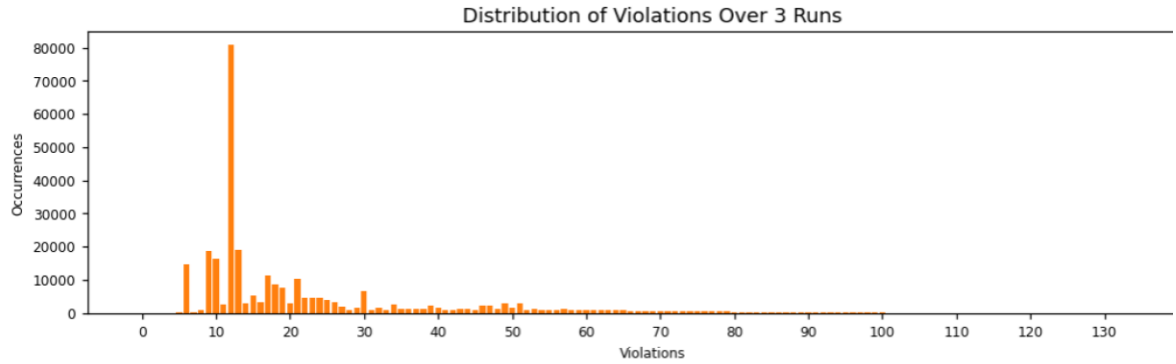



File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
Penalized fitness histogram:
Distribution of Penalized Fitnesses Over 3 Runs
10000
0
0 25 50 75 100 125 150 175 200 225 250 275 300 325 350
Penalized Fitness (Rounded)
Base fitness histogram:
Distribution of Base Fitnesses Over 3 Runs
300000
299997

ile Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
Violation histogram:
```



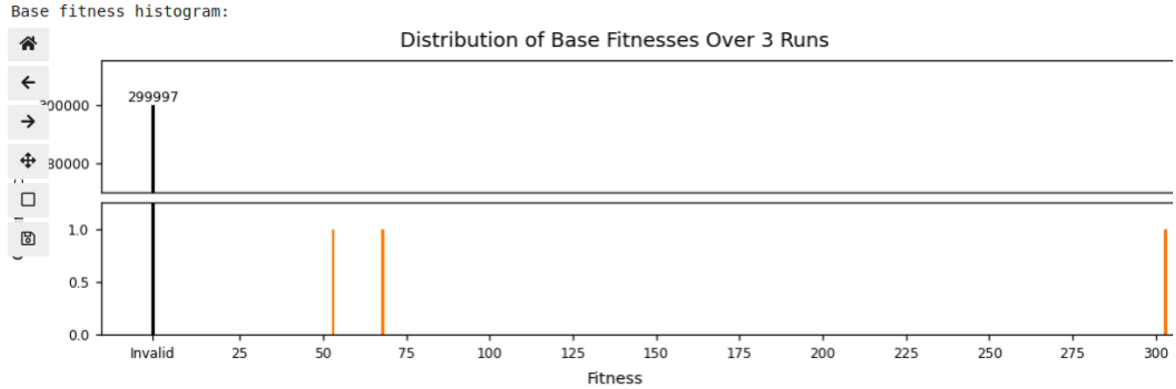
Just like in previous assignments, we've provided a function to save your experimental data for you:

[36]:
```
def save_data(best_per_run, best_solution, mean_data, max_data, mean_base_data, max_base_data, \
              valid_data, eval_counts, hist, base_hist, violation_hist, logs, subdir, config):
    subdir.mkdir(parents=True, exist_ok=True)

    with open(subdir / 'best_per_run.txt', 'w') as f:
```

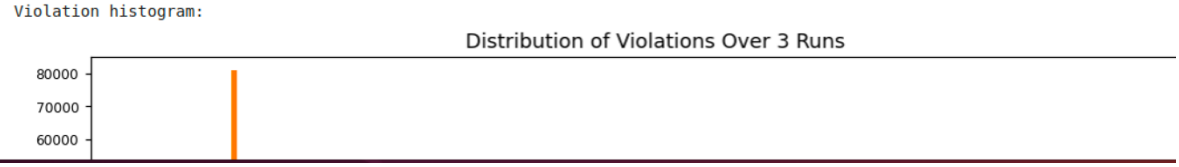
ile Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

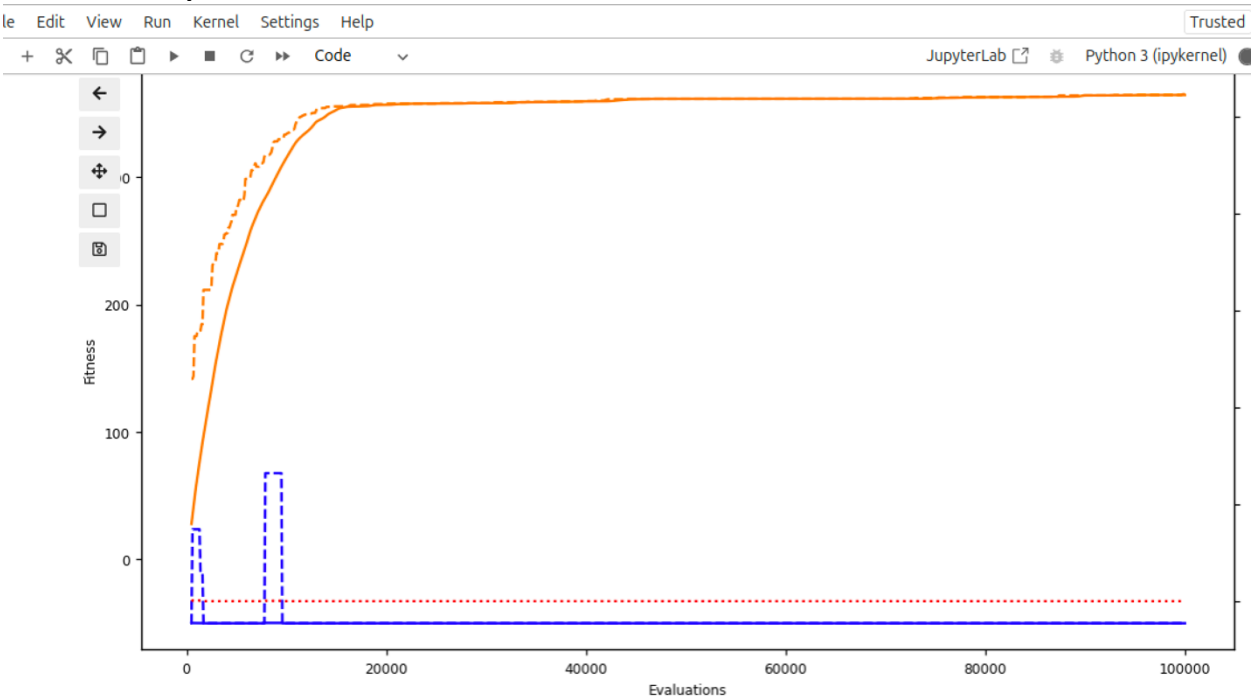
```
Base fitness histogram:
```



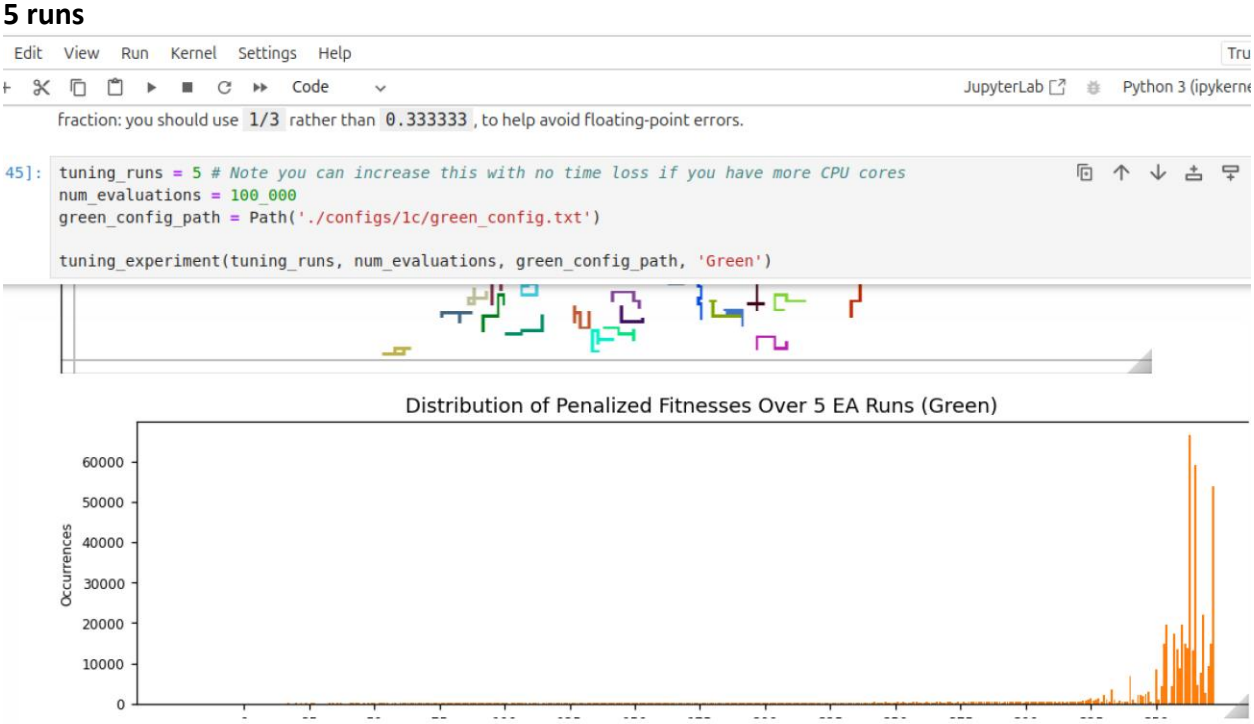

```
Violation histogram:
```



Eval v Fitness plot



5 runs

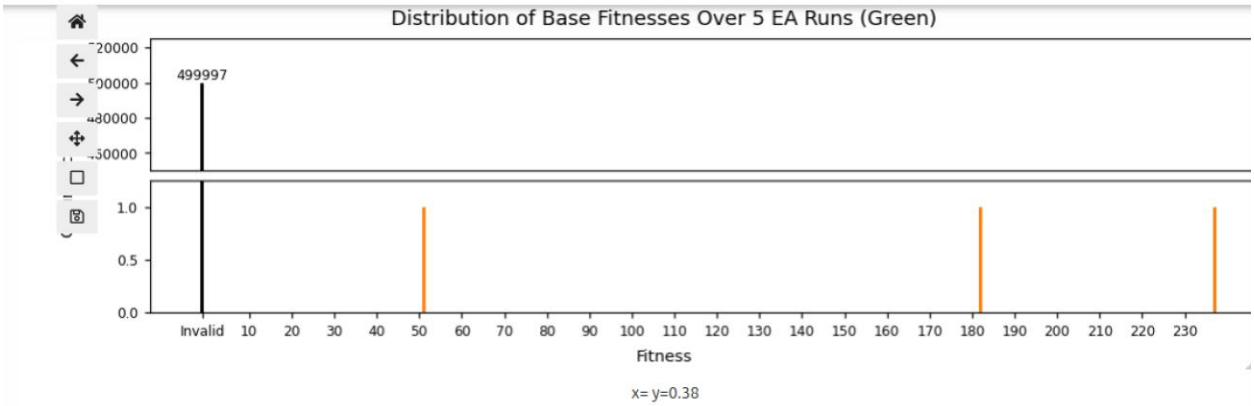


Edit View Run Kernel Settings Help

Code JupyterLab Python 3 (ipykerne

fraction: you should use `1/3` rather than `0.333333`, to help avoid floating-point errors.

45]:
```
tuning_runs = 5 # Note you can increase this with no time loss if you have more CPU cores
num_evaluations = 100_000
green_config_path = Path('./configs/1c/green_config.txt')

tuning_experiment(tuning_runs, num_evaluations, green_config_path, 'Green')
```



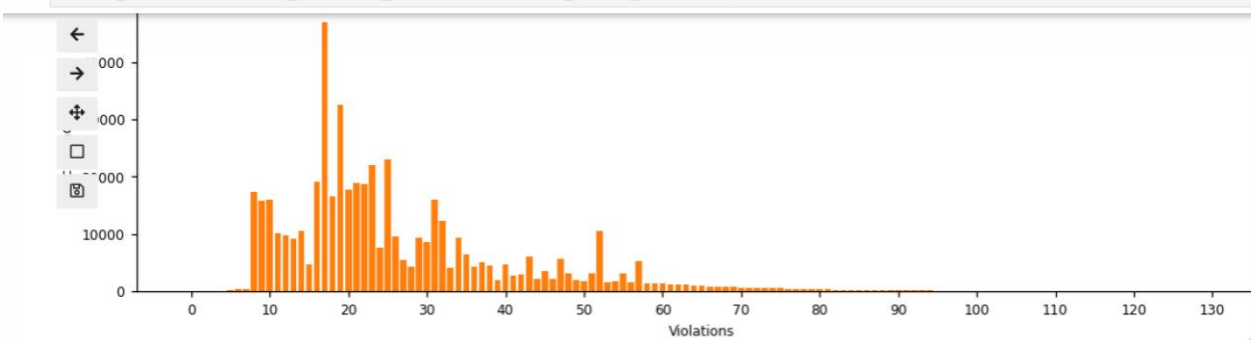
Distribution of Violations Over 5 EA Runs (Green)

Edit View Run Kernel Settings Help

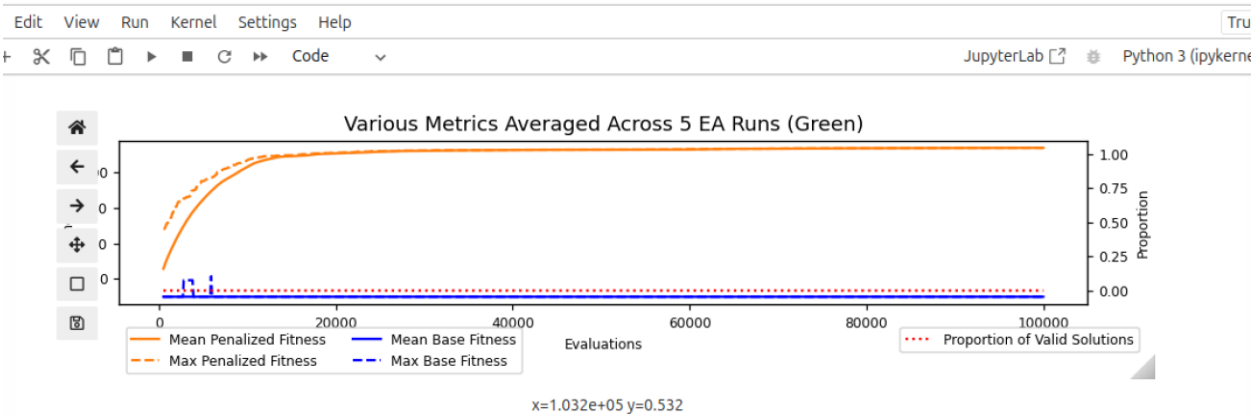
Code JupyterLab Python 3 (ipykerne

fraction: you should use `1/3` rather than `0.333333`, to help avoid floating-point errors.

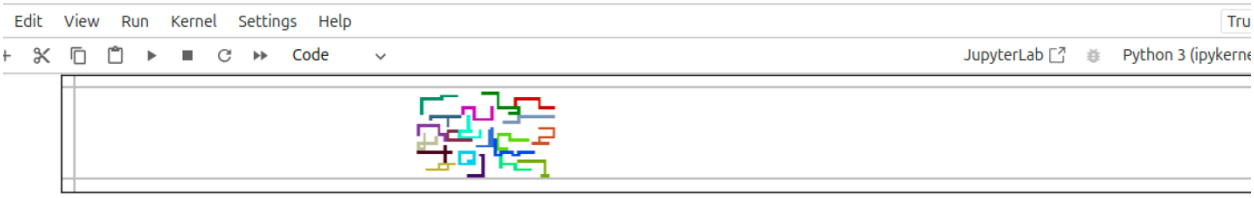
45]:
```
tuning_runs = 5 # Note you can increase this with no time loss if you have more CPU cores
num_evaluations = 100_000
green_config_path = Path('./configs/1c/green_config.txt')

tuning_experiment(tuning_runs, num_evaluations, green_config_path, 'Green')
```

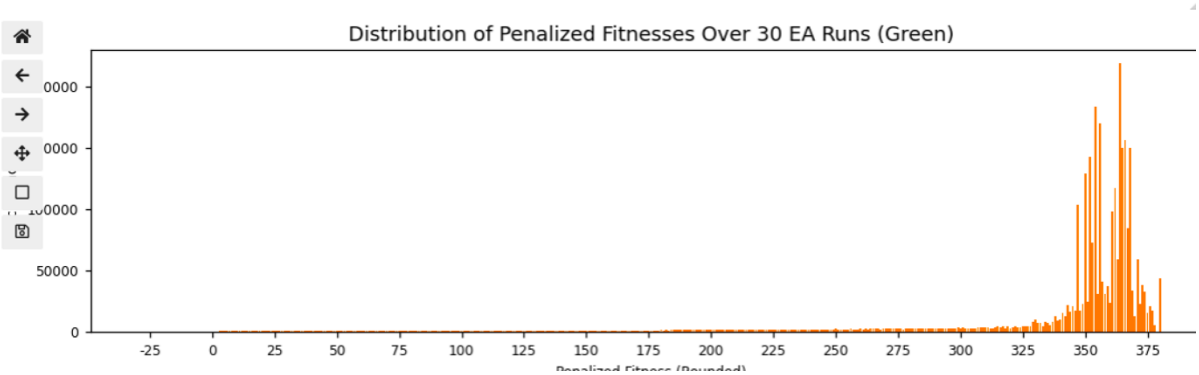


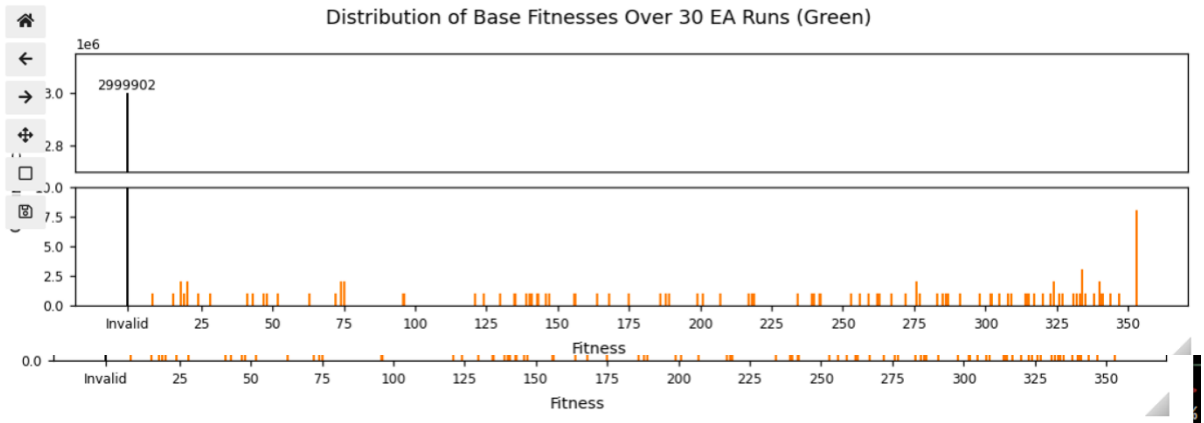
Various Metrics Averaged Across 5 EA Runs (Green)

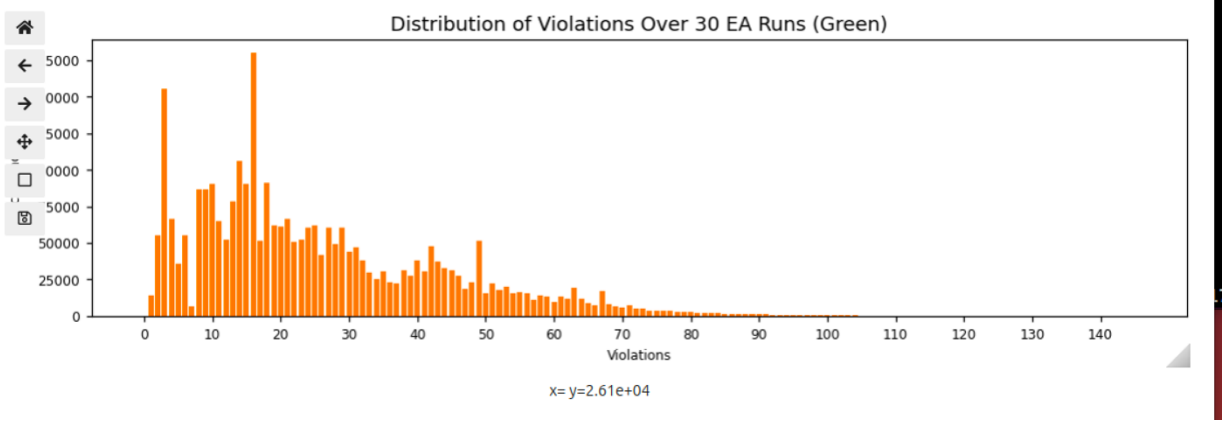

Edit View Run Kernel Settings Help
Code
JupyterLab
Python 3 (ipykerne
Various Metrics Averaged Across 5 EA Runs (Green)
1.00
0.75
0.50
0.25
0.00
Proportion
0
20000
40000
60000
80000
100000
Evaluations
Mean Penalized Fitness
Max Penalized Fitness
Mean Base Fitness
Max Base Fitness
Proportion of Valid Solutions
x=1.032e+05 y=0.532

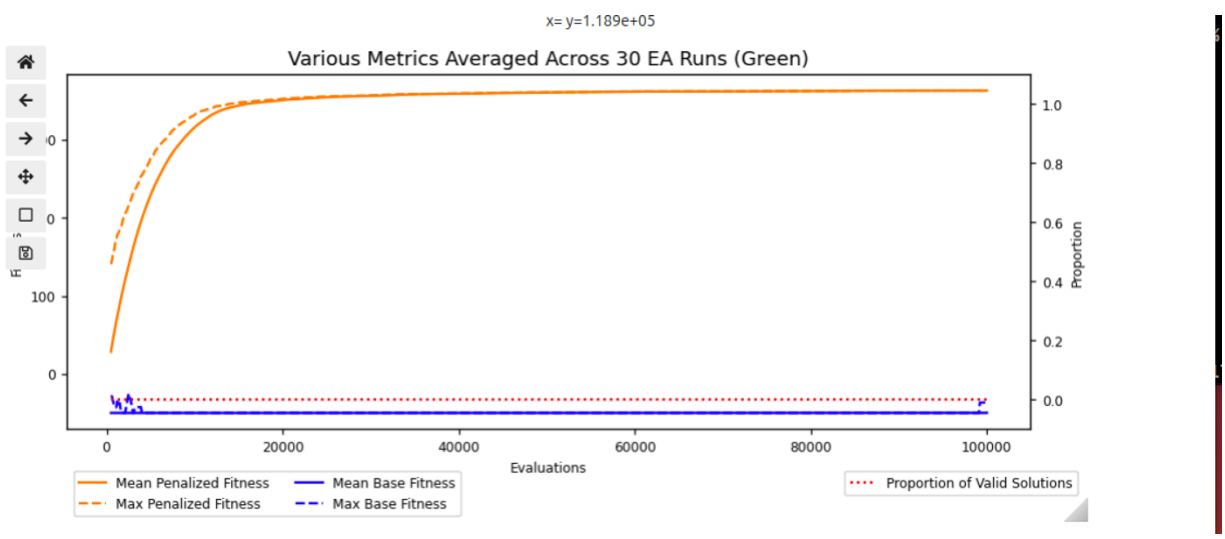
Over 30 runs


Edit View Run Kernel Settings Help
Code
JupyterLab
Python 3 (ipykerne


Distribution of Penalized Fitnesses Over 30 EA Runs (Green)
50000
0
-25
0
25
50
75
100
125
150
175
200
225
250
275
300
325
350
375







Average Penalized Fitness (Final): 28.969
Best Penalized Fitness: 209.34375

Average Base Fitness (Final): -49.97451
Best Base Fitness: 88
Analysis: The plot shows an increase in penalized fitness over generations
Best Base Fitness: 88, indicating some solutions are relatively strong
Violations Histogram skewed toward zero since there are "31 valid solutions
The number of valid solutions and the penalized fitness distribution reveal that the EA can find high quality solutions

Best Penalized Fitness
Assignment 1b:
Hard Green:
Maximum Best Fitness: 98.4 over 5 runs.
Average Best Fitness: 81.53 over 30 runs.
Hard Yellow:
Maximum Best Fitness: 318.
Hard Red:
Maximum Best Fitness: 296.
Assignment 1c:
Maximum Best Fitness: 209.34375.
Average Base Fitness: 88.
Analysis:
In Assignment 1b, the best fitness values in the Hard configurations were generally higher than in Assignment 1c especially for Hard Yellow and Hard Red.
In Assignment 1c, the best penalized fitness is 209.34375, which is higher than the best fitness for Hard Green in Assignment 1b (98.4). it is lower than the best fitness values for Hard Yellow 318 and Hard Red 296 in Assignment 1b.

the algorithm in Assignment 1c performed well, it didn't surpass the best cases in some Hard configurations of Assignment 1b, especially in variants Yellow and Red

Average Penalized Fitness
Assignment 1b:

Hard Green:
Average: 81.53.
Variation: 173.45.

Hard Yellow:
Mean Best Fitness: 72.67.
Standard Deviation: 212.46.

Hard Red:
Mean Best Fitness: 65.33.
Standard Deviation: 199.76.

Assignment 1c:
Average Penalized Fitness : 28.969.
Average Base Fitness: -49.97451.

Analysis:
The average penalized fitness in Assignment 1c (28.969) is lower than the averages seen in the various configurations of Assignment 1b. In 1b, the averages for Hard Green, Hard Yellow, and Hard Red are all above 65.
The large negative average base fitness in 1c suggests that most solutions in the population struggled with the problem.
The high variations in 1b, particularly for Hard Yellow (212.46) and Hard Red (199.76) indicate that the algorithm's performance was inconsistent, finding both high and lowquality solutions.

Assignment 1c managed to find a larger number of valid solutions (31) while assignment 1 b did not
Assignment 1c is better in handling constraints and found more valid solutions unlke 1B
1C has Lower average penalized fitness compared to Assignment 1b