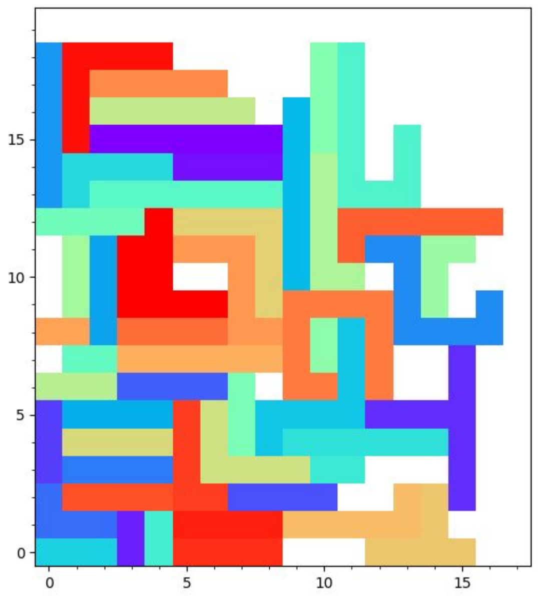
Multi-Objective Evolutionary Algorithms for Stock Cutting

Michael Harrington, mth347@mst.edu

Computer Science CS 5401 Fall 2017 Assignment 1D

Attempting All Bonus



Solution File	Fitness Details	Parent Selection	Survival Selection	Other Config Details	Initial Population	Assignment Requirement	Label
1	Length, Width	Random	Truncation	parents = 20 offspring = 140 comma adaptive mutations	Random	General	general_f1
2	Length, Width	Random	Truncation	parents = 20 offspring = 140 comma adaptive mutations	Random	General	general_f2
3	Length, Width	Random	Truncation	parents = 20 offspring = 140 comma adaptive mutations	Random	General	general_f3
1	Length, Width, Adjacents	Random	Truncation	parents = 20 offspring = 140 comma adaptive mutations	Random	Bonus 1 Part 1	bonus_1_p1_f1
1	Length	Random	Truncation	parents = 20 offspring = 140 comma adaptive mutations	bonus_1_p1_f1	Bonus 1 Part 2	bonus_1_p2_f1
2	Length, Adjacents	Random	Truncation	parents = 20 offspring = 140 comma adaptive mutations	Random	Bonus 1 Part 2	bonus_1_p2_f2_s1
2	Length	Random	Truncation	parents = 20 offspring = 140 comma adaptive mutations	bonus_1_p2_f2_s1	Bonus 1 Part 2	bonus_1_p2_f2_s2
1	Length, Width	Random	Crowding	parents = 20 offspring = 140 comma adaptive mutations	Random	Bonus 2 Part 1	bonus_2_p1_f1
1	Length, Width	Random	Truncation with Sharing	parents = 20 offspring = 140 comma adaptive mutations	Random	Bonus 2 Part 2	bonus_2_p2_f1
1	Length, Width, Adjacents	Random	Crowding	parents = 20 offspring = 140 comma adaptive mutations	Random	Bonus 2 Part 3	bonus_2_p3_f1
1	Length	Random	Crowding	parents = 20 offspring = 140 comma adaptive mutations	bonus_1_p1_f1	Bonus 2 Part 4	bonus_2_p4_f1
1	Cut	Random	Truncation	parents = 20 offspring = 140 comma adaptive mutations	Random	Bonus 3	bonus_3_f1_s1
1	Length, Cut	Random	Truncation	parents = 20 offspring = 140 comma adaptive mutations	Random	Bonus 3	bonus_3_f1_s2
1	Length	Random	Truncation	parents = 20 offspring = 140 comma adaptive mutations	bonus_3_f1_s2	Bonus 3	bonus_3_f1_s3

Introduction

The multi-objective evolutionary algorithm analyzed here has a tunable config file and creates solutions to arranging shapes on a roll of stock. Arranging these shapes can save material per product, and the importance of utilizing this resource well is paramount to many industrial applications. While the conventional stock cutting problem is one of optimizing length of stock, this evolutionary algorithm explores optimizing other objectives simultaneous. These additional objectives include maximizing adjacent edges (minimizing cut), minimizing use of stock width, and minimizing adjacent shapes.

Genome Representation

In contrast to previous assignments, this genome representation has changed to a simpler direct mapping to the phenotype. The dominant crossover picks a gene for each shape template from one of the parents. To mimic dominant genes, it is slightly more likely to pick genes closer to the left side of the stock. This genome representation has seemed to alleviate the plateau effect we saw in earlier assignments. However, in addition for more exploration, there is a chance offspring will be generated using the permutation crossover. This crossover uses adjacency relationships to create offspring. Its explorative power has proved useful in the beginning of searches - it may prove useful in strange multi-objective environments as well. This crossover ratio is adaptable and we find it usually adapts down to near 0 over time.

Note

The data in this report will be shown with two graphs for each fitness metric. The primary graph is box plots of the best fitness over 30 runs over time, while the second is the average fitness box plotted with a step line of the best fitness. Notice the y axis for determining the fitness metric the graph is measuring. Additionally, note that parts are meant to refer to distinct subsections that count for bonus points while steps are results of different configs created and ran to complete that bonus.

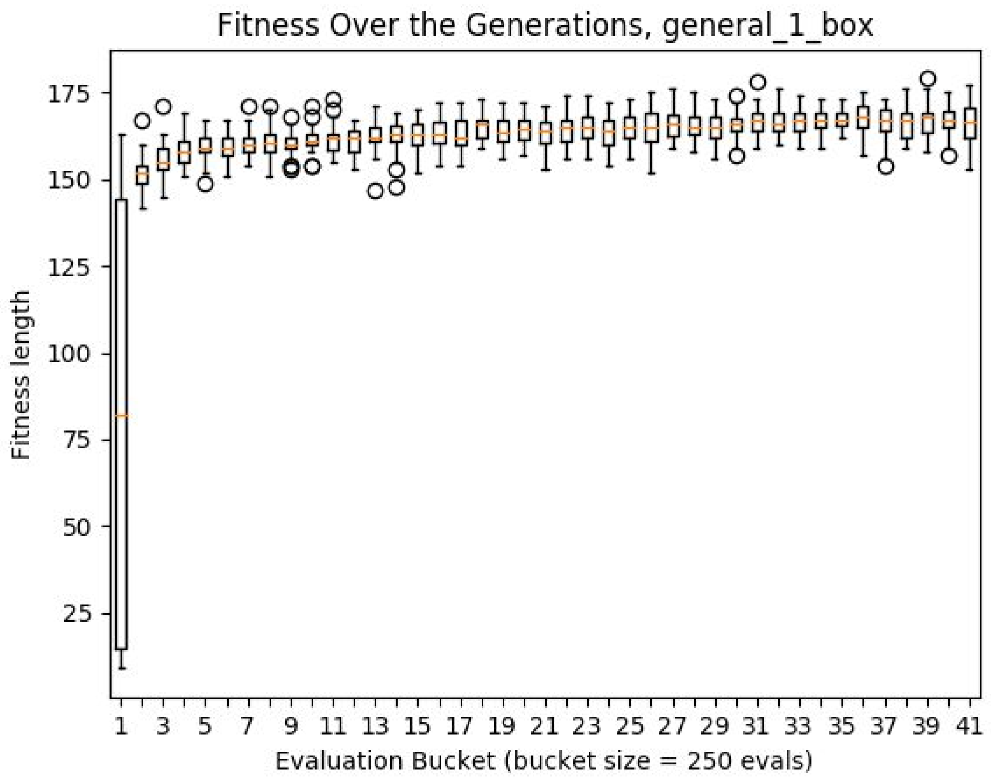
Initial Experiment

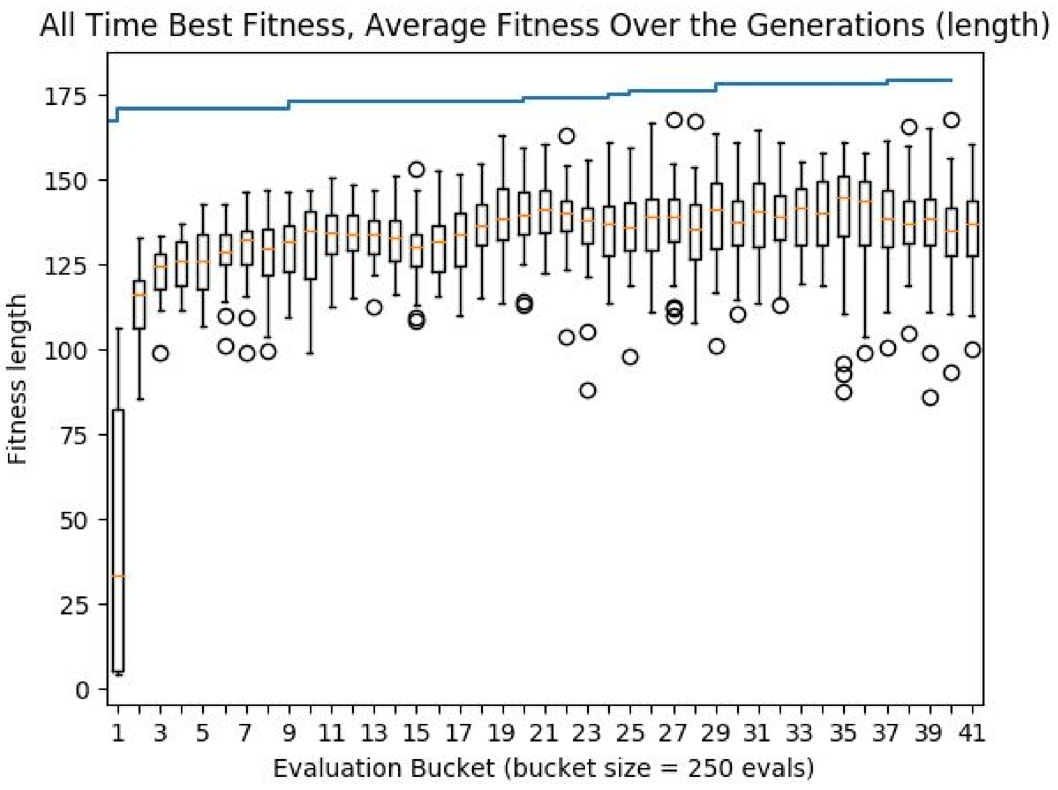
The initial experiment explores the three problem files with multi-objectives of length and width. These can be seen in the reference table as general_1, general_2, and general_3. The results of these are seen below.

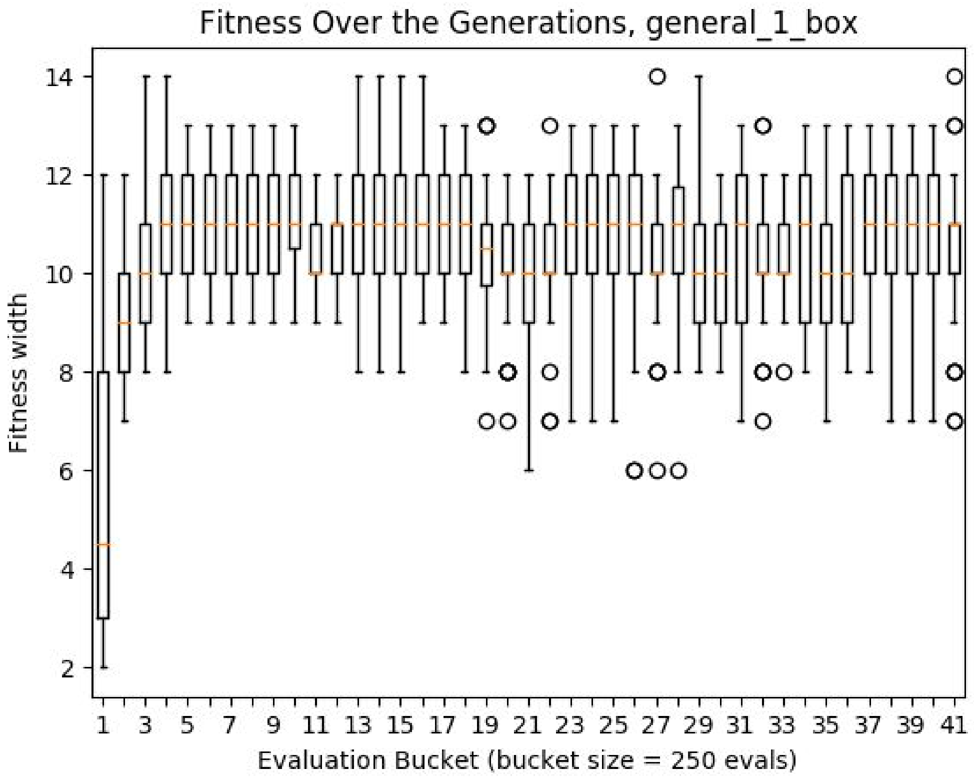
General

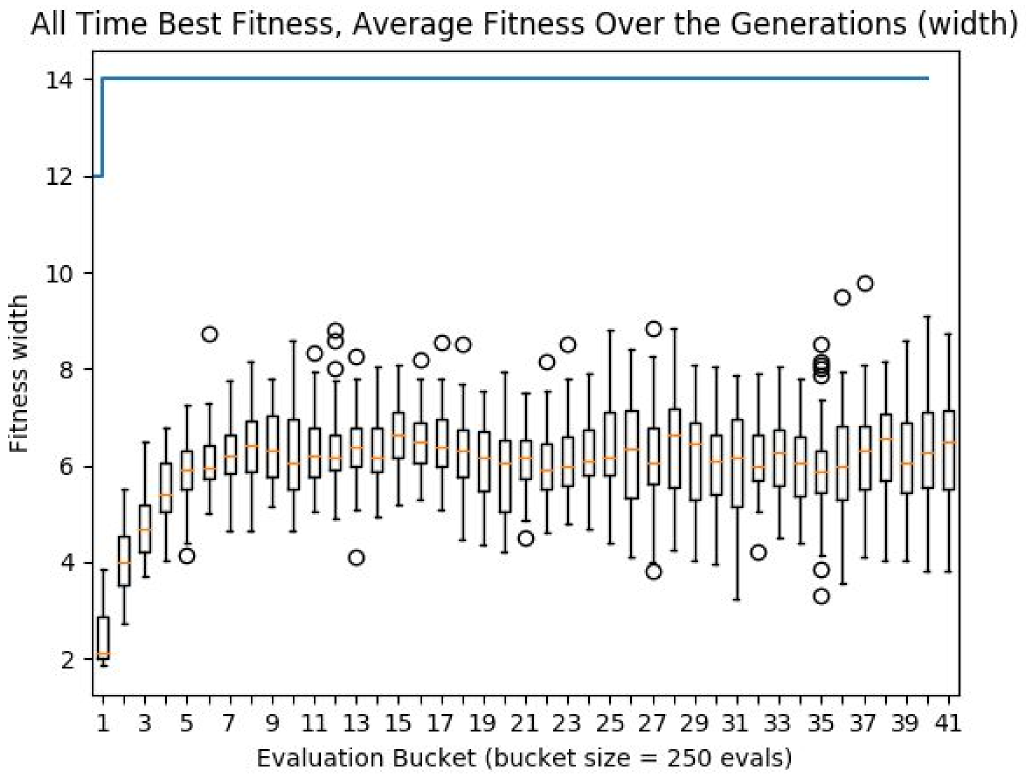
general_1

This config plateaus in the length fitness quite rapidly, along with the width fitness. While the length fitness has appreciable results, the width fitness is mediocre. It appears these fitness measures may contradict too much to provide good results over time. While the config does well for single fitness values, perhaps it suffers from advantages in genetic diversity of large populations.



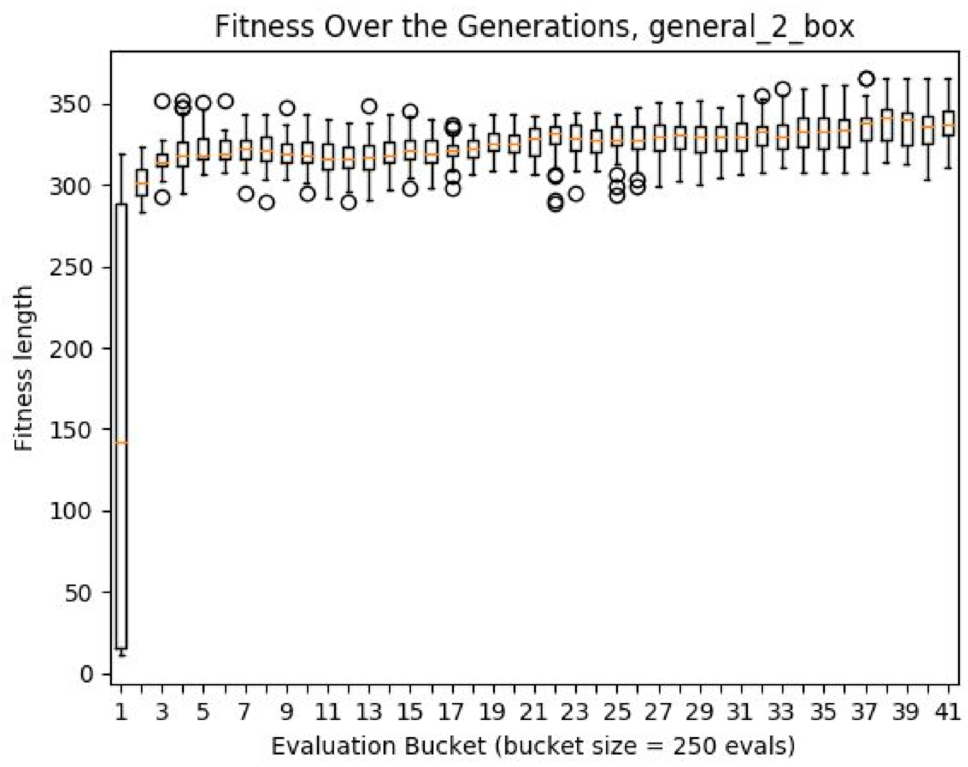

All Time Best Fitness, Average Fitness Over the Generations (length)
Fitness length
Evaluation Bucket (bucket size = 250 evals)
0
25
50
75
100
125
150
175
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41

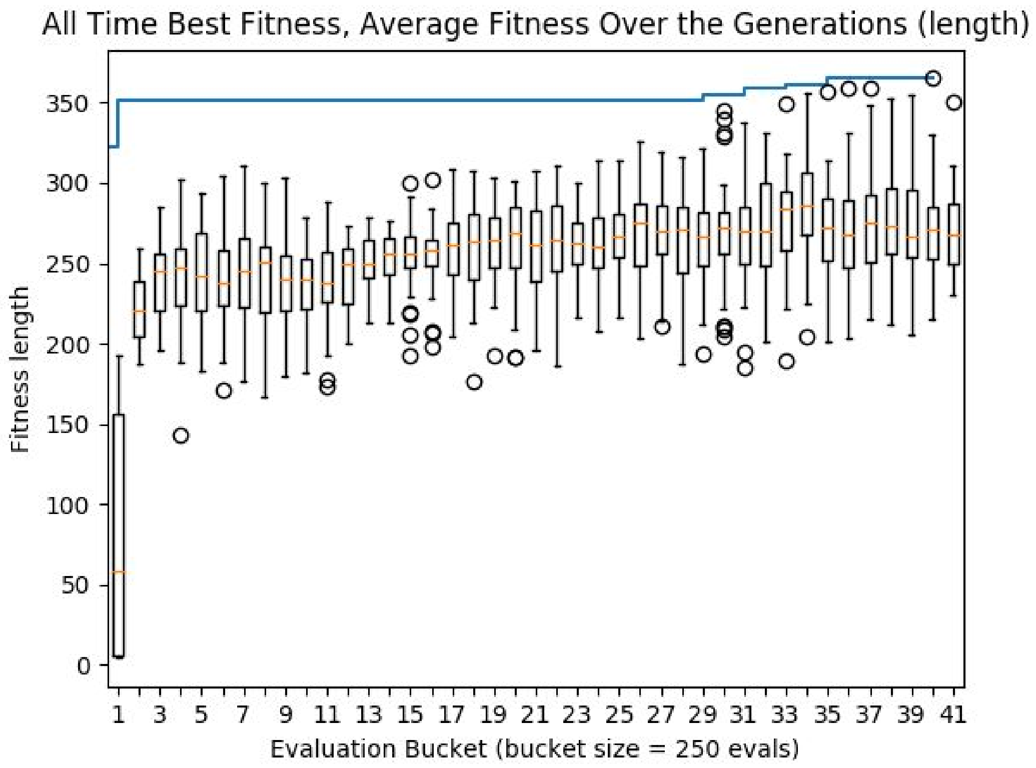

Fitness Over the Generations, general_1_box
Fitness width
Evaluation Bucket (bucket size = 250 evals)
2
4
6
8
10
12
14
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41

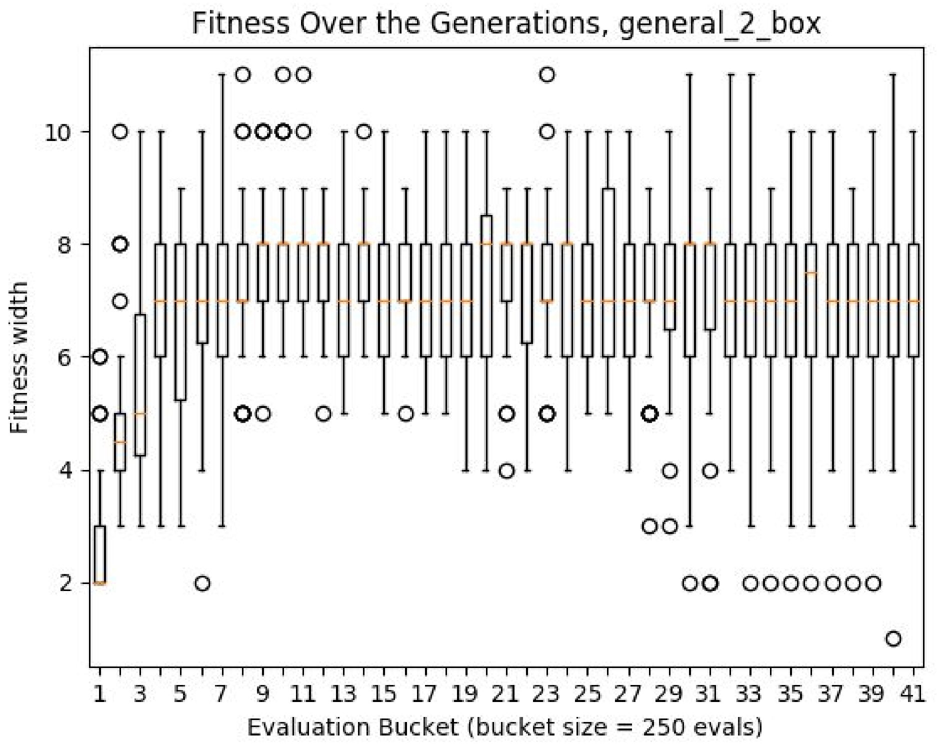

All Time Best Fitness, Average Fitness Over the Generations (width)
Fitness width
14
12
10
8
6
4
2
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
Evaluation Bucket (bucket size = 250 evals)

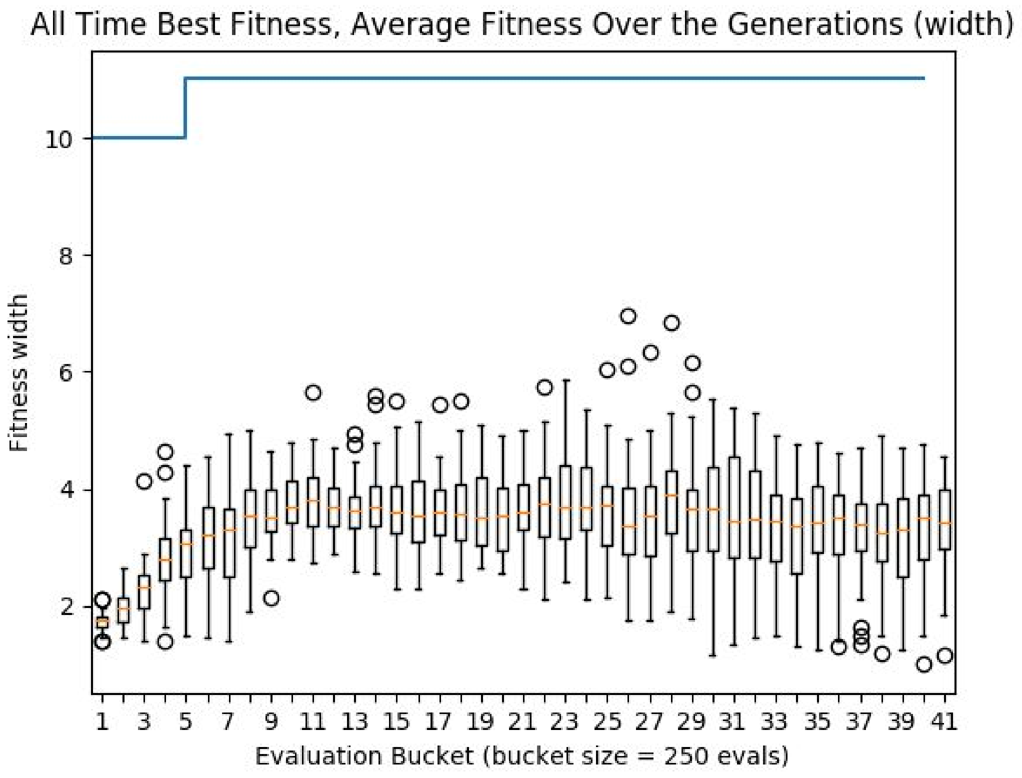
general_2

We see similar results here, a quick plateau with slight improvement on the length fitness and a really variadic width fitness. The sharp rises in the beginning are due to the permutation crossover talked about earlier.



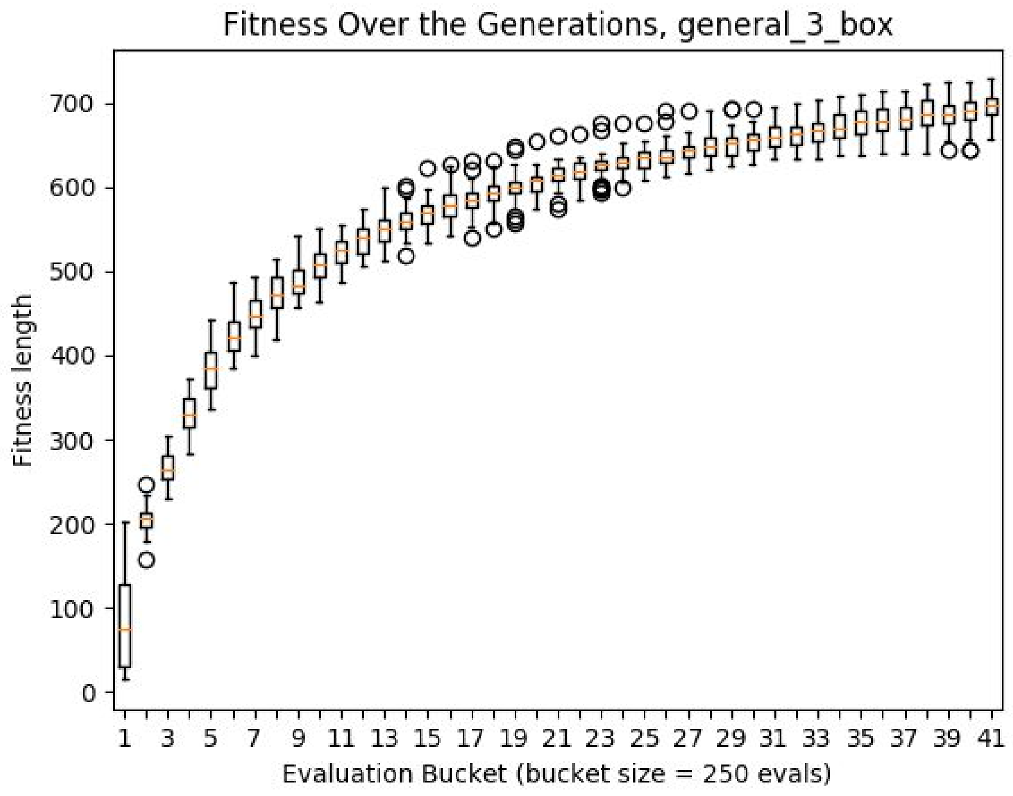


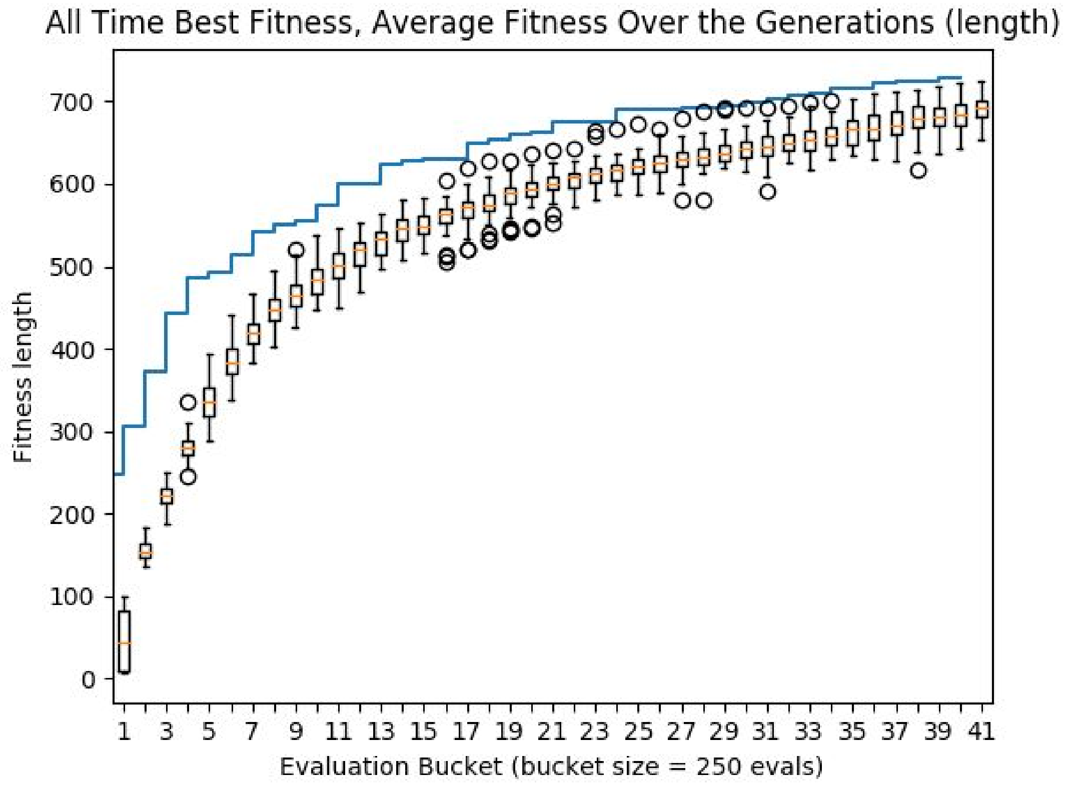

Fitness Over the Generations, general_2_box
Fitness width
Evaluation Bucket (bucket size = 250 evals)

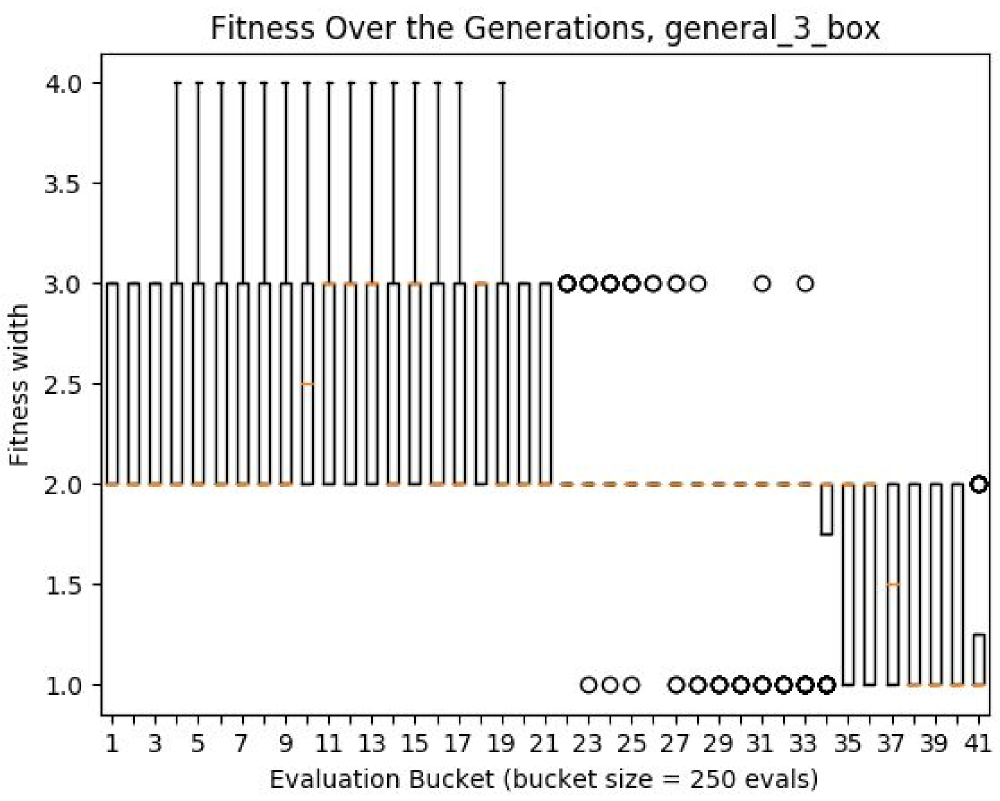

All Time Best Fitness, Average Fitness Over the Generations (width)
Fitness width
Evaluation Bucket (bucket size = 250 evals)

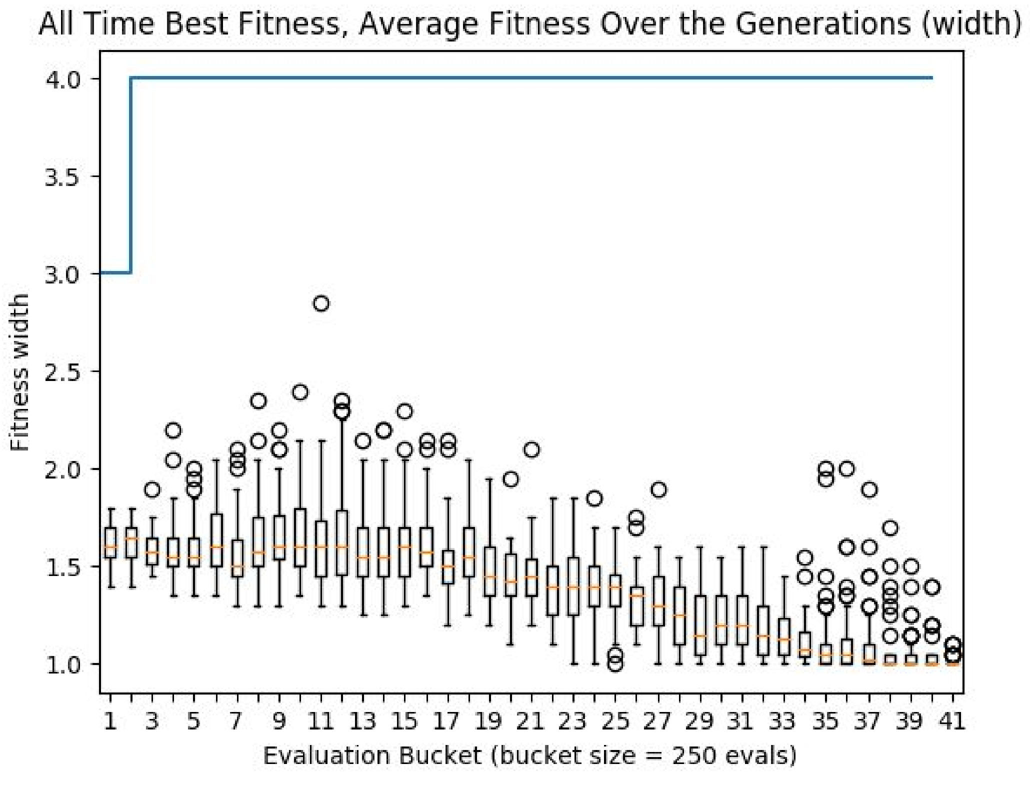
general_3

Interestingly, the third shape file does not improve well on the width and has nice progression on the length.






Fitness Over the Generations, general_3_box
Fitness width
Evaluation Bucket (bucket size = 250 evals)
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41

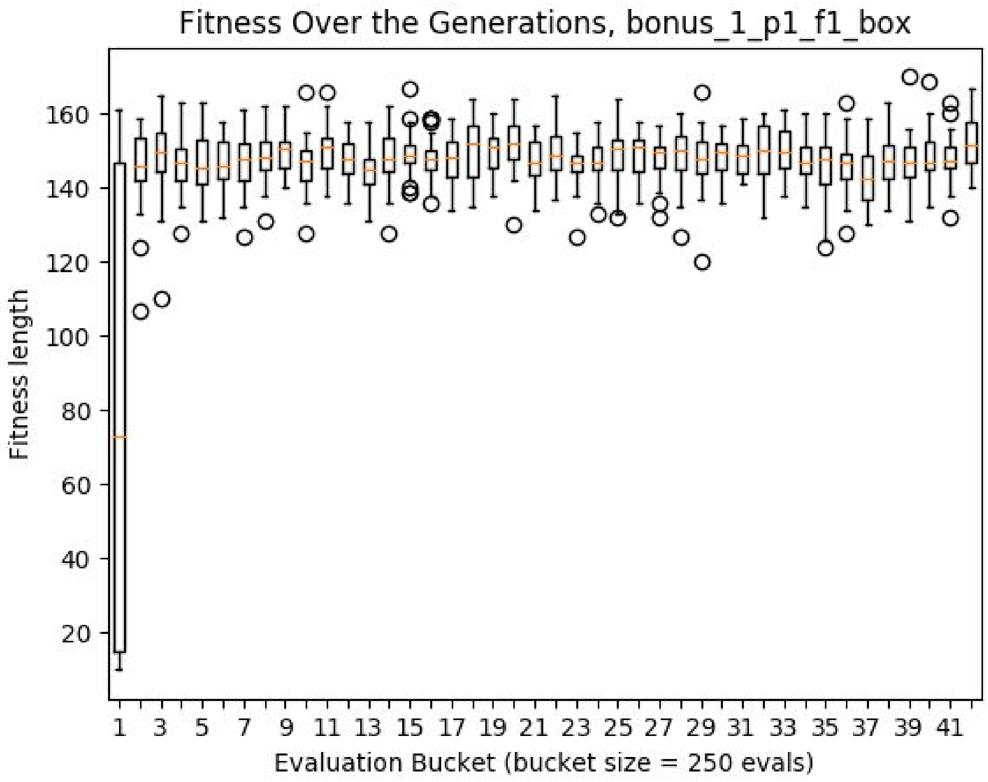

All Time Best Fitness, Average Fitness Over the Generations (width)
Fitness width
Evaluation Bucket (bucket size = 250 evals)
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41

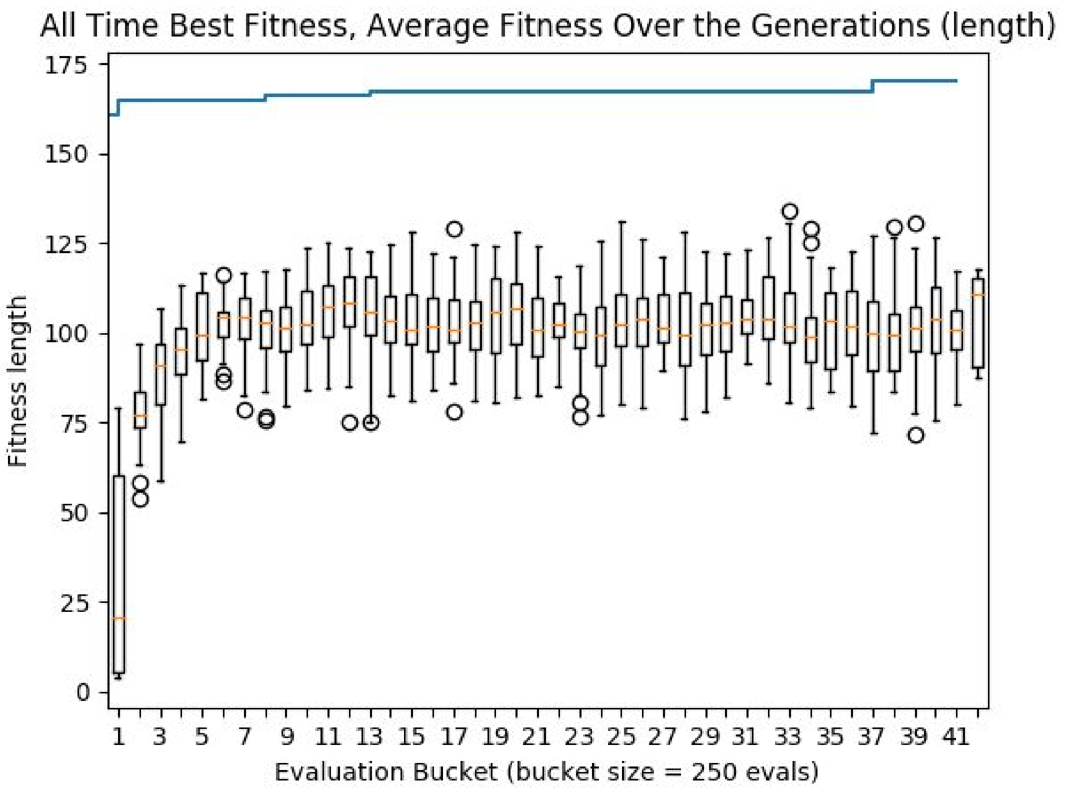
Bonus 1

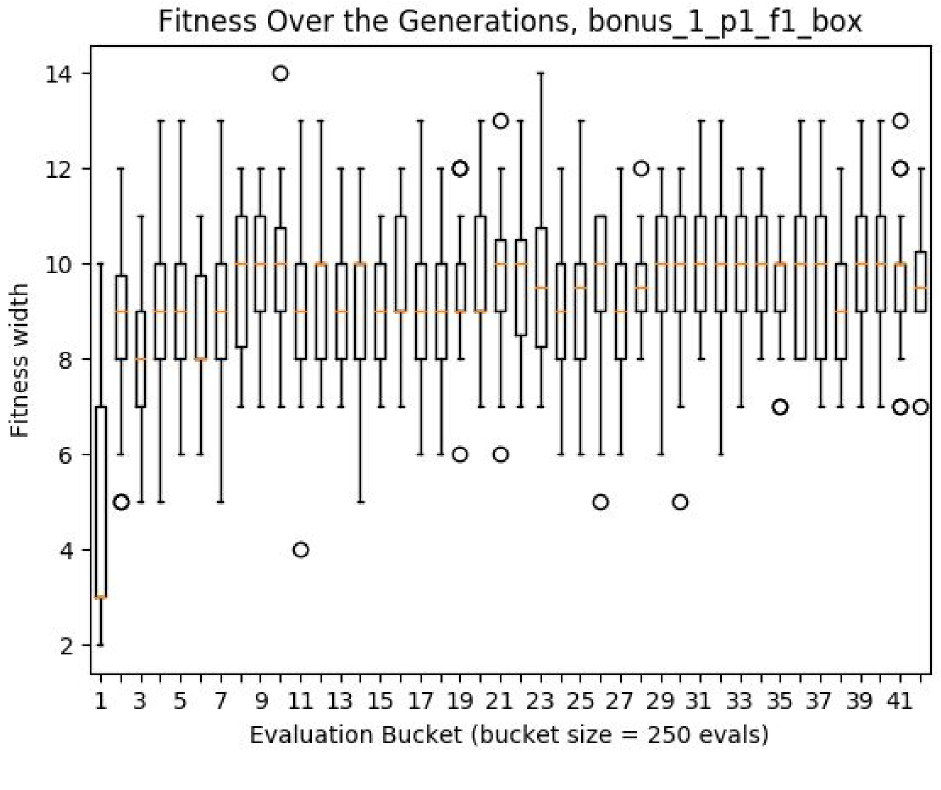
part 1

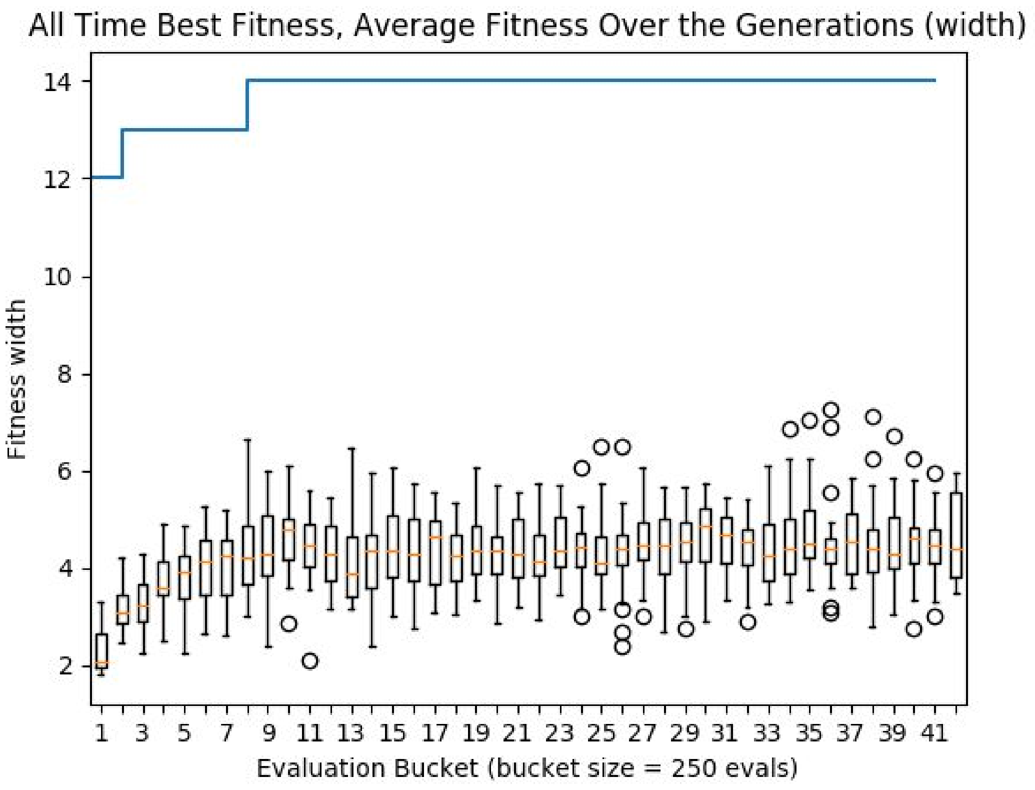
To expand the exploration of multi-objective evolutionary algorithms, this explores a third very conflicting objective: reducing the number of adjacent shapes. The idea here is to prevent the shapes from premature convergence and explore orientations before limiting their freedom in movement. Then use that solution file for initialization of another run. So as we would expect, the length fitness measure did statistically worse than the counterpart in the 2 objective ea explored earlier.

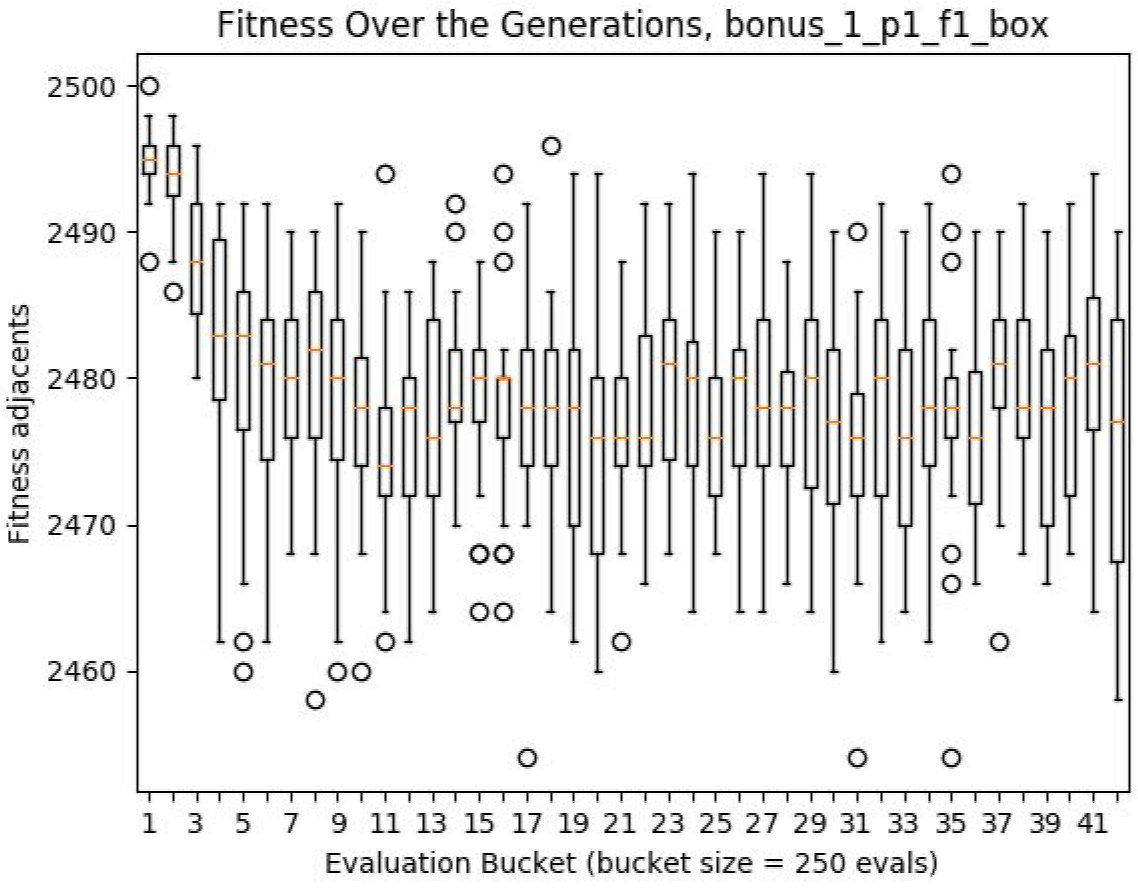
Note that the graph of the adjacent fitness may look interesting, but this fitness measure is max number of adjacencies minus the number of adjacencies. We also see that a random solution starts with this at max and due to the other objectives, the average only goes down from there.

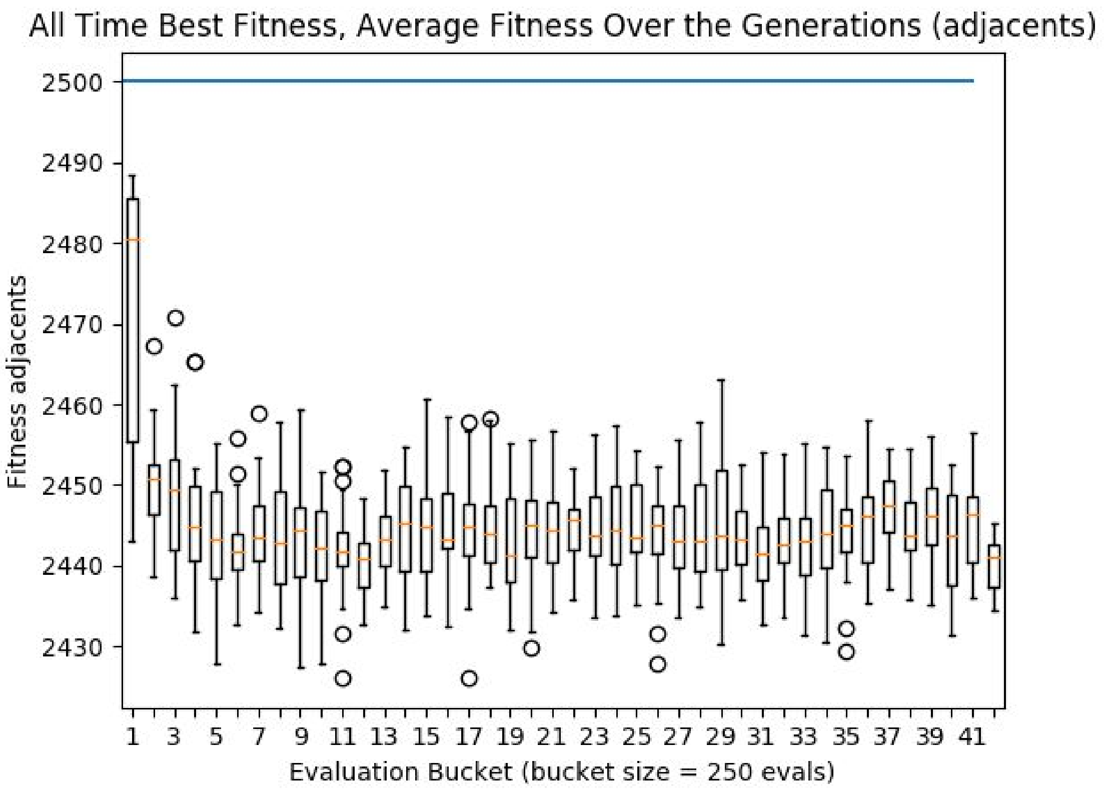



All Time Best Fitness, Average Fitness Over the Generations (length)
Fitness length
Evaluation Bucket (bucket size = 250 evals)
0
25
50
75
100
125
150
175
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41


Fitness Over the Generations, bonus_1_p1_f1_box
Fitness width
Evaluation Bucket (bucket size = 250 evals)
2
4
6
8
10
12
14
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41

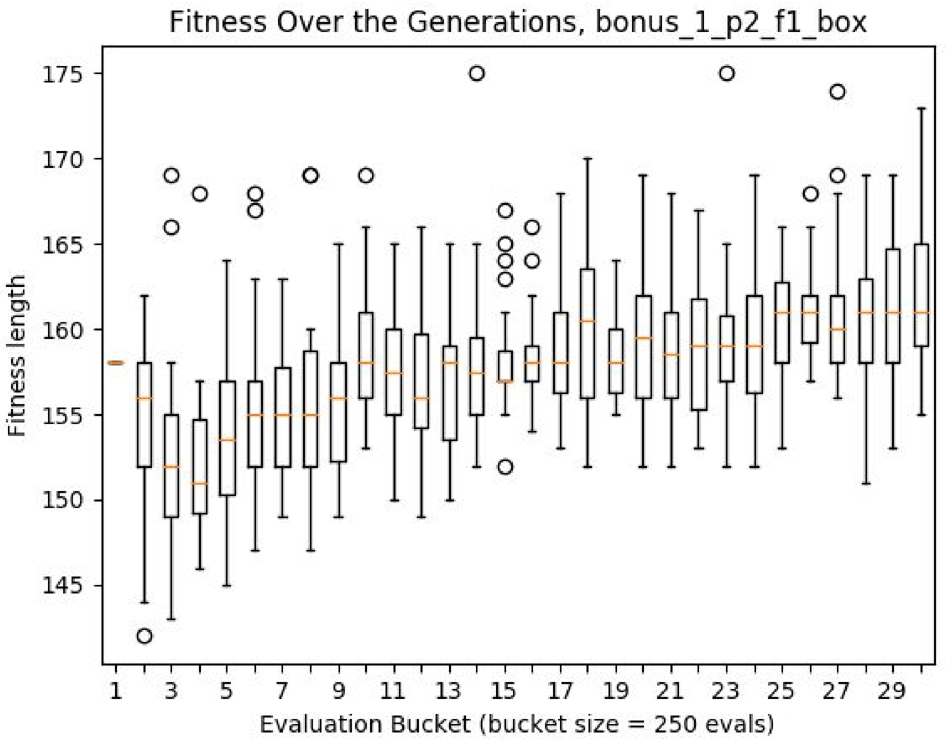

All Time Best Fitness, Average Fitness Over the Generations (width)
Fitness width
Evaluation Bucket (bucket size = 250 evals)
2
4
6
8
10
12
14
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41

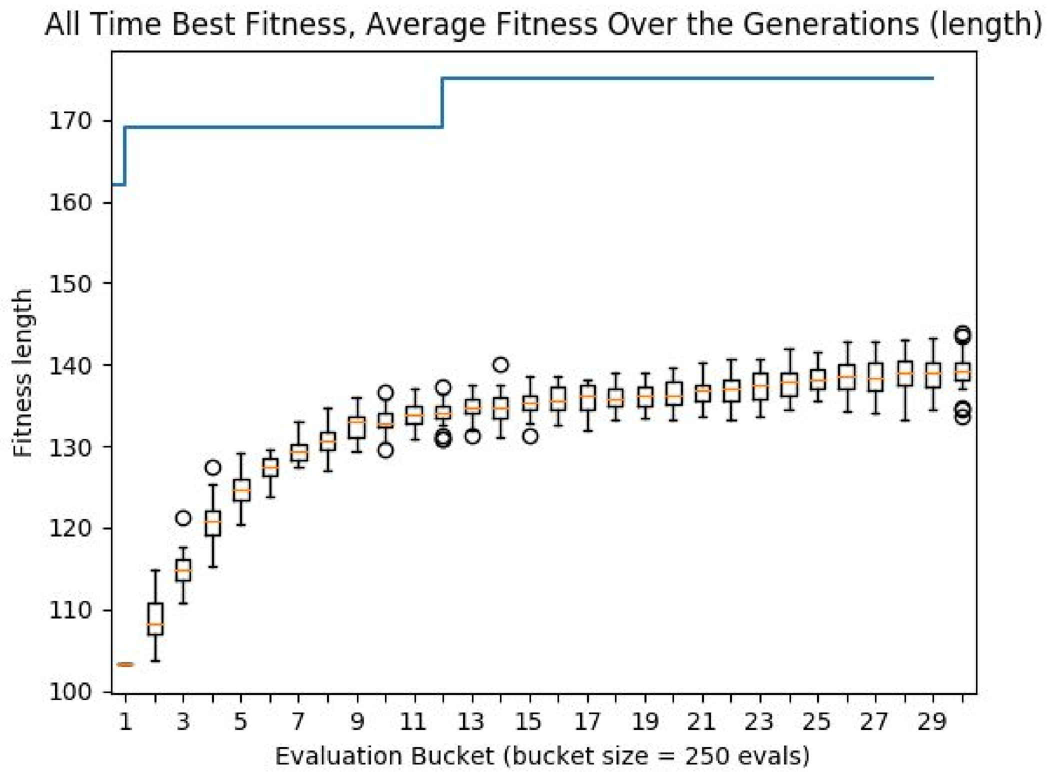

Fitness Over the Generations, bonus_1_p1_f1_box
Fitness adjacents
Evaluation Bucket (bucket size = 250 evals)
2460
2470
2480
2490
2500
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41



part 2, problem 1

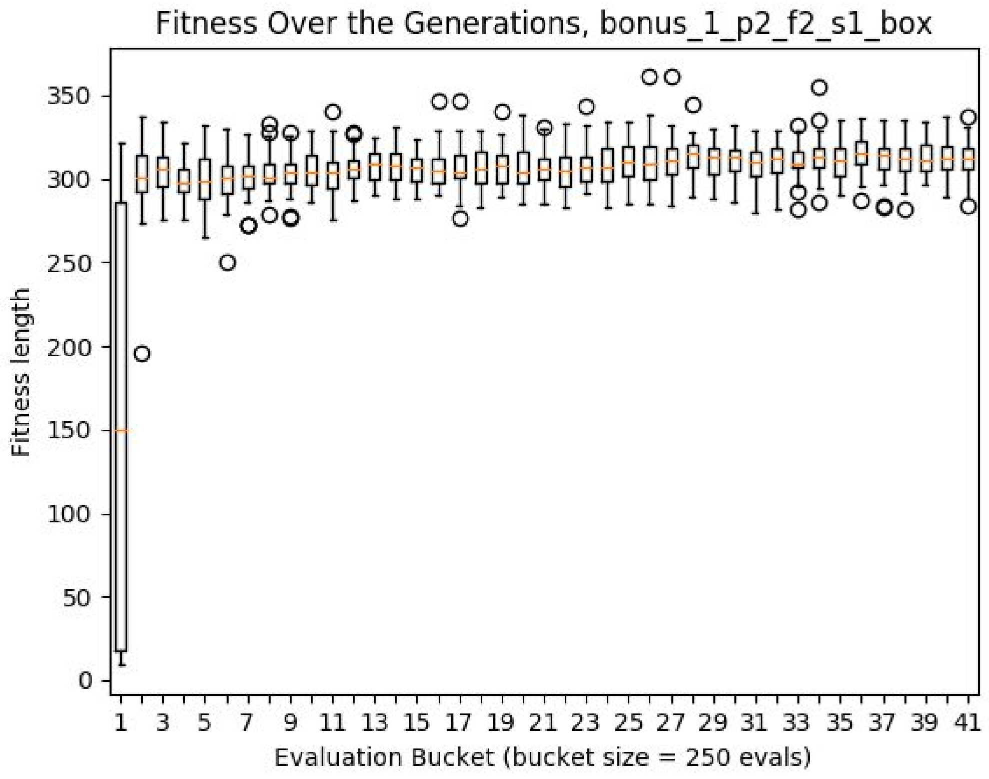
To continue the exploration of this additional objective, the previous experiment's output solution are used as input to the EA with just length as the only objective. Disappointingly it does statistically worse than the MOEA with length and width as objectives which did more with less evaluations.

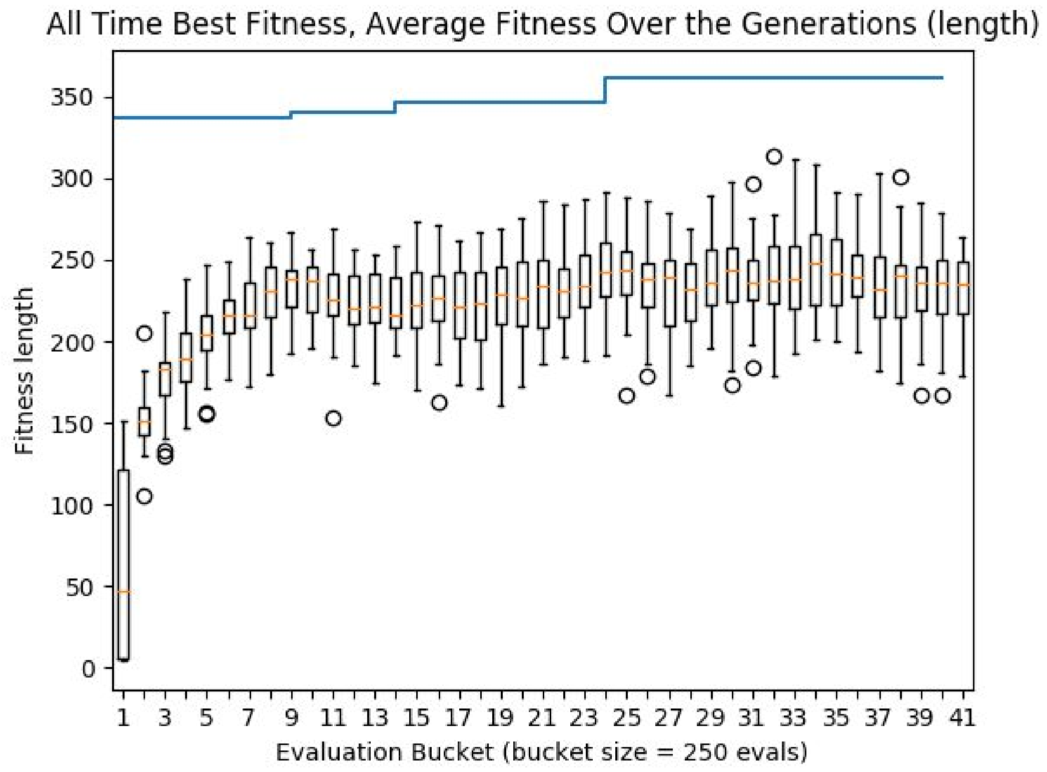


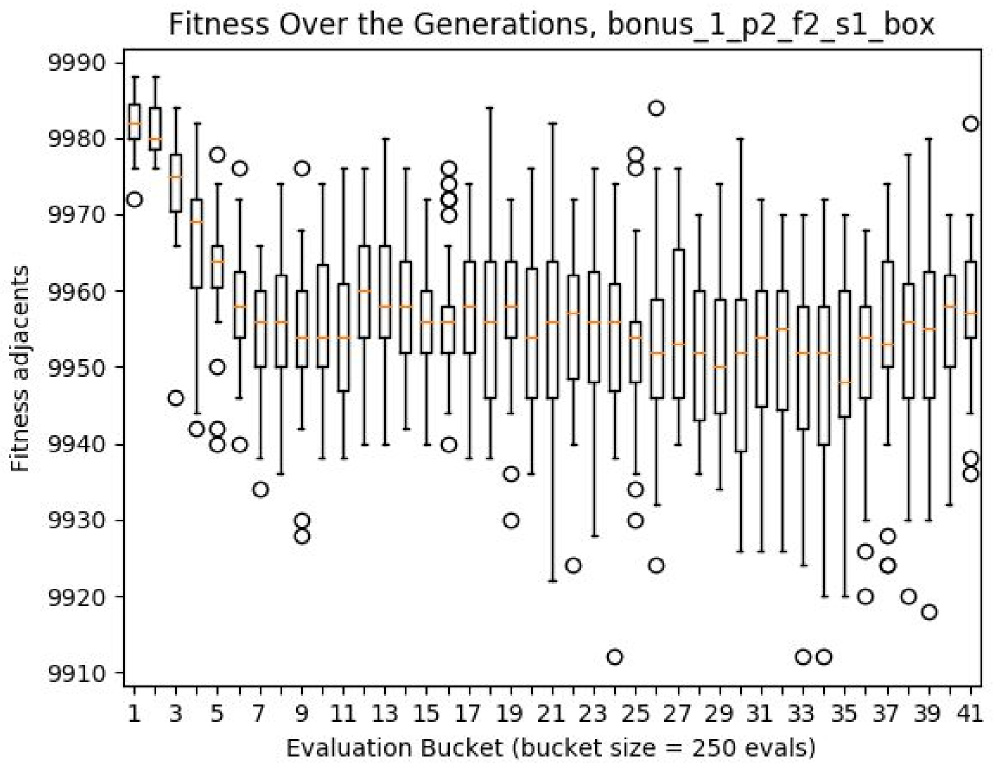


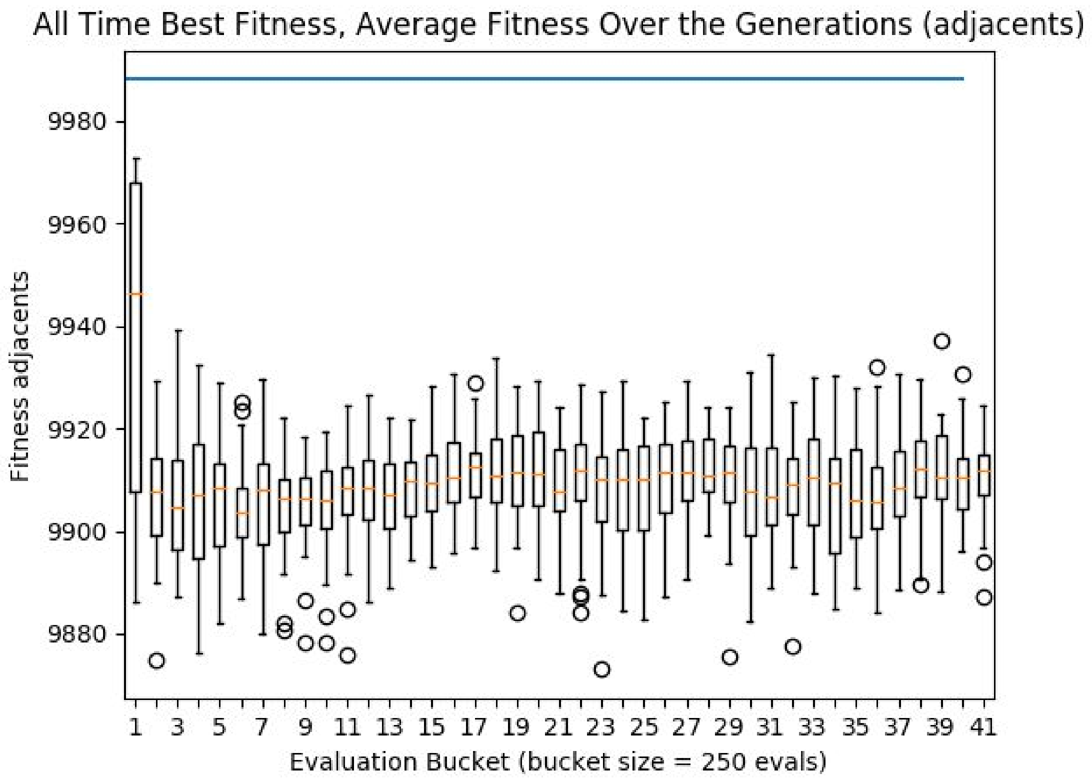
part 2, problem 2, step 1

To not give up on the adjacent fitness objective, I decided to give it a try but without using width in the initial experiment. This is the first experiment with length and adjacents as the objective.



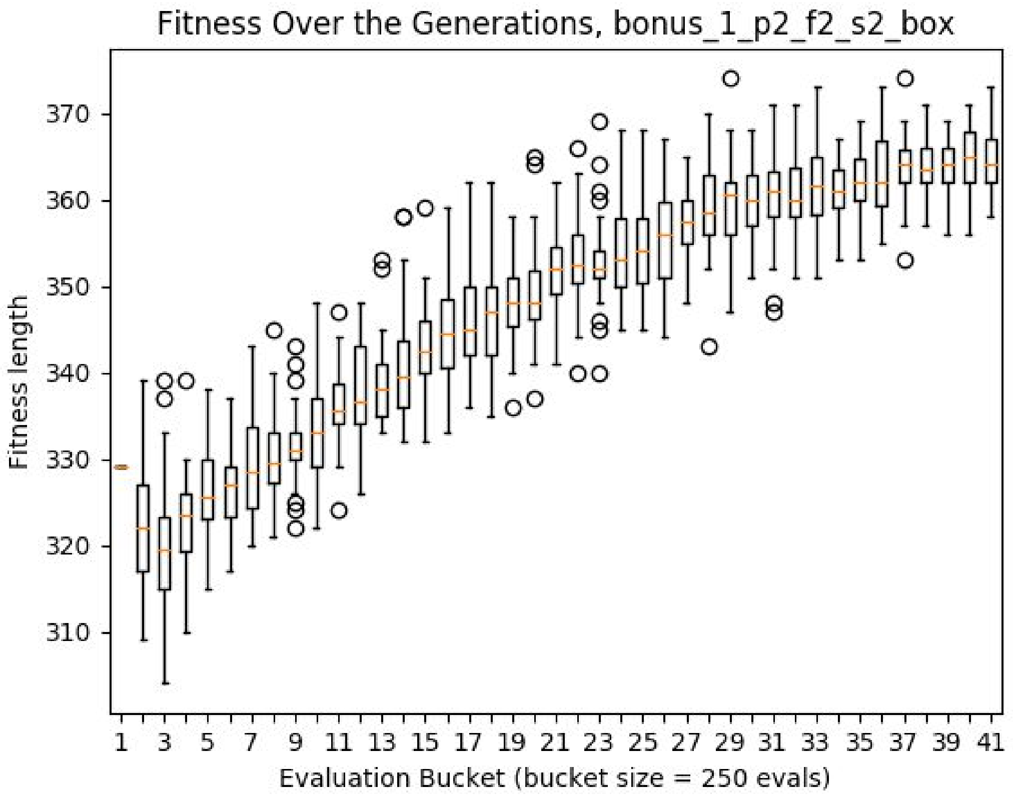

All Time Best Fitness, Average Fitness Over the Generations (length)
Fitness length
Evaluation Bucket (bucket size = 250 evals)
0
50
100
150
200
250
300
350
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41

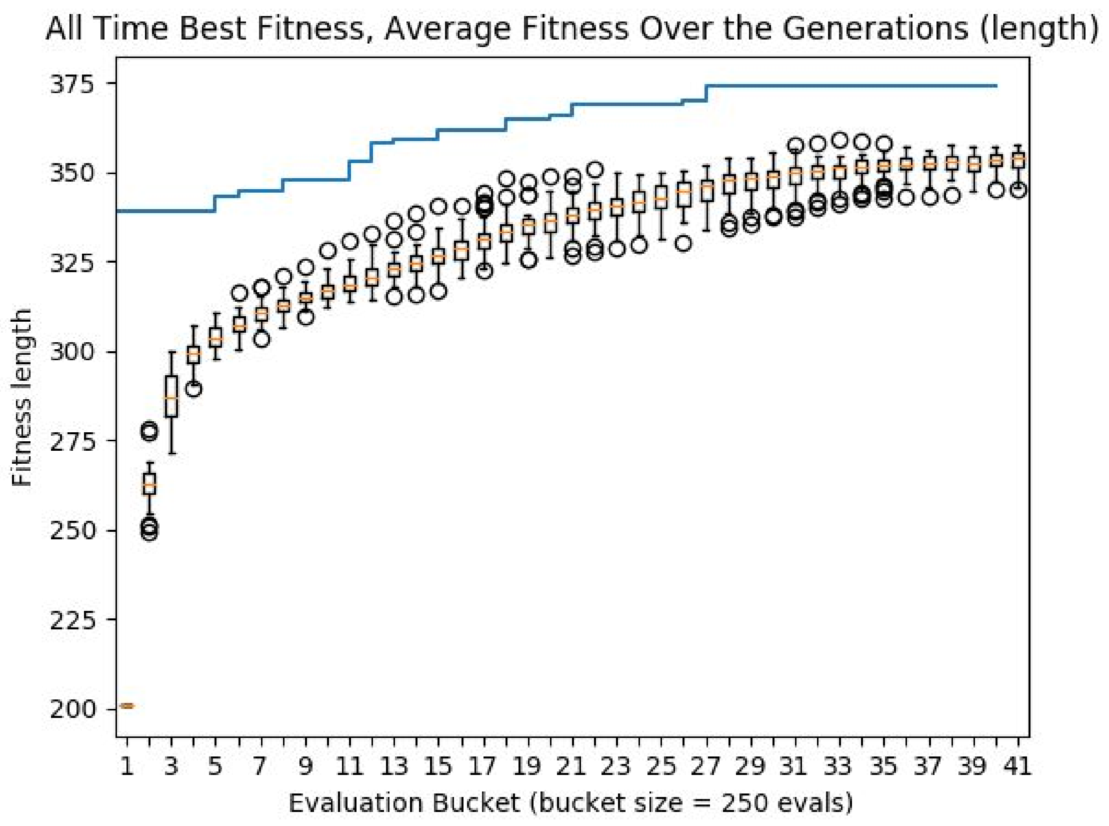

Fitness Over the Generations, bonus_1_p2_f2_s1_box
Fitness adjacents
Evaluation Bucket (bucket size = 250 evals)
9910
9920
9930
9940
9950
9960
9970
9980
9990
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41



part 2, problem 2, step 2

This is the continuation of the experiment and you can see tha with problem file 2 this initial seeding allowed it to perform better than the general_2 counterpart which plateaued at much lower length fitness.

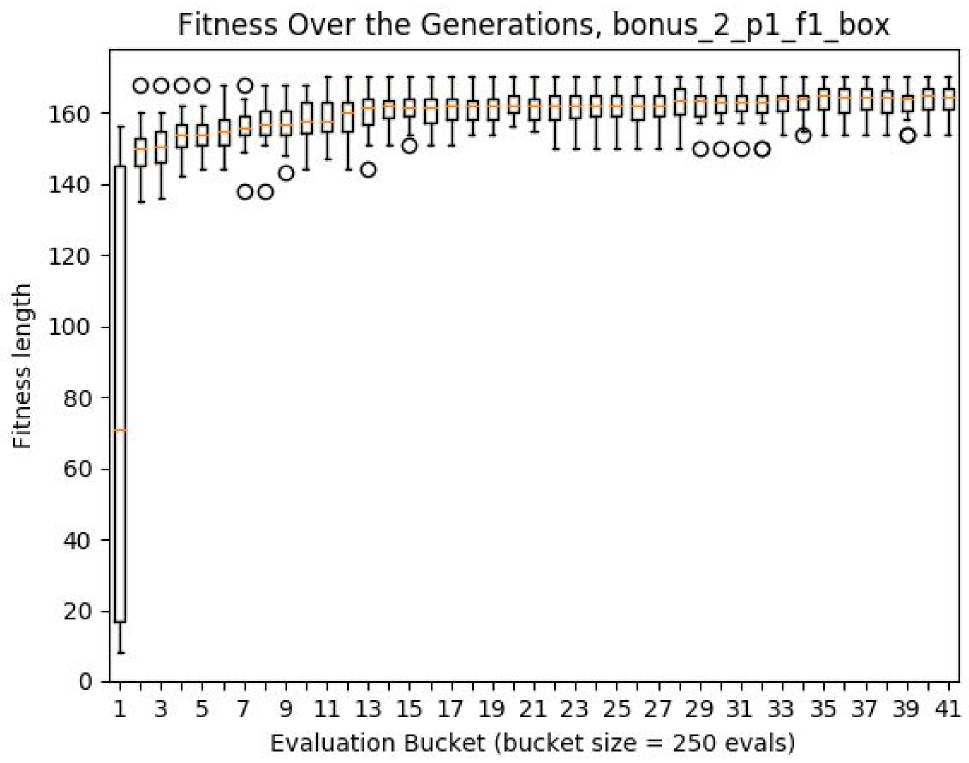


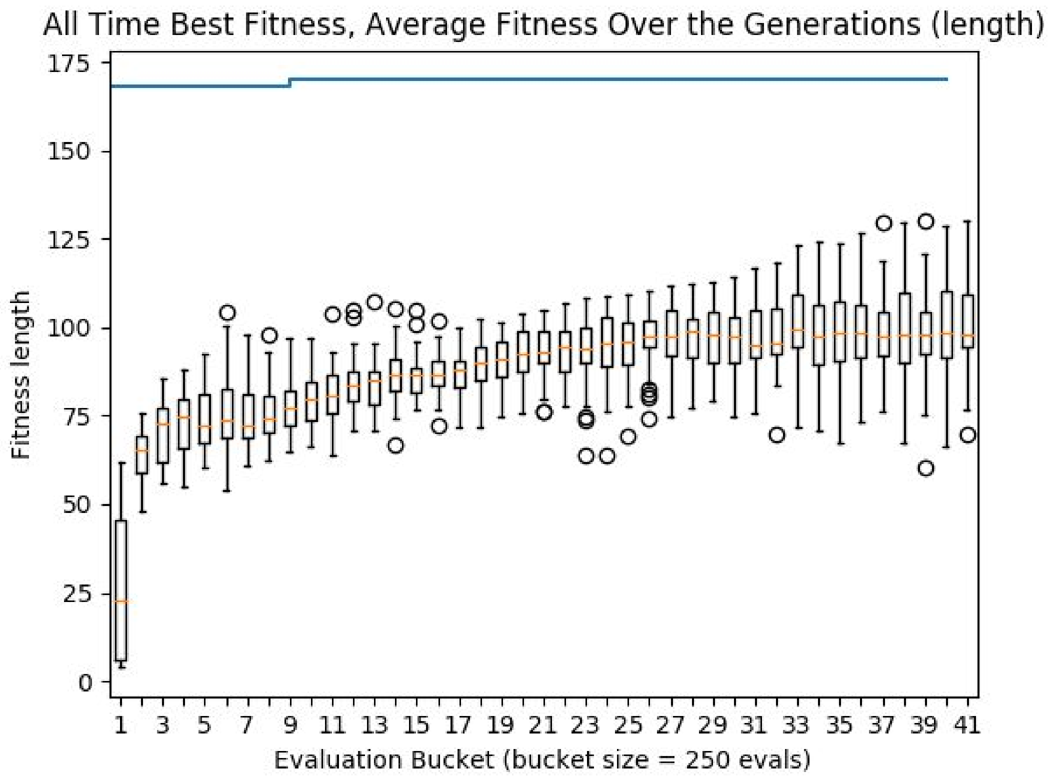

All Time Best Fitness, Average Fitness Over the Generations (length)
Fitness length
Evaluation Bucket (bucket size = 250 evals)
200
225
250
275
300
325
350
375
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41

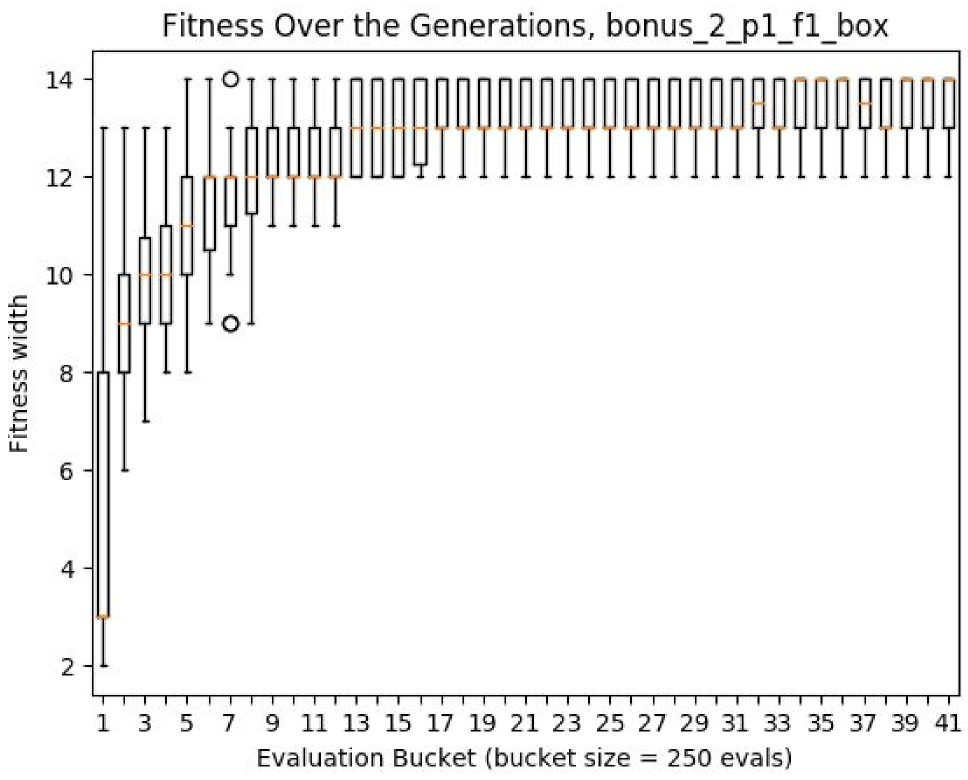
Bonus 2

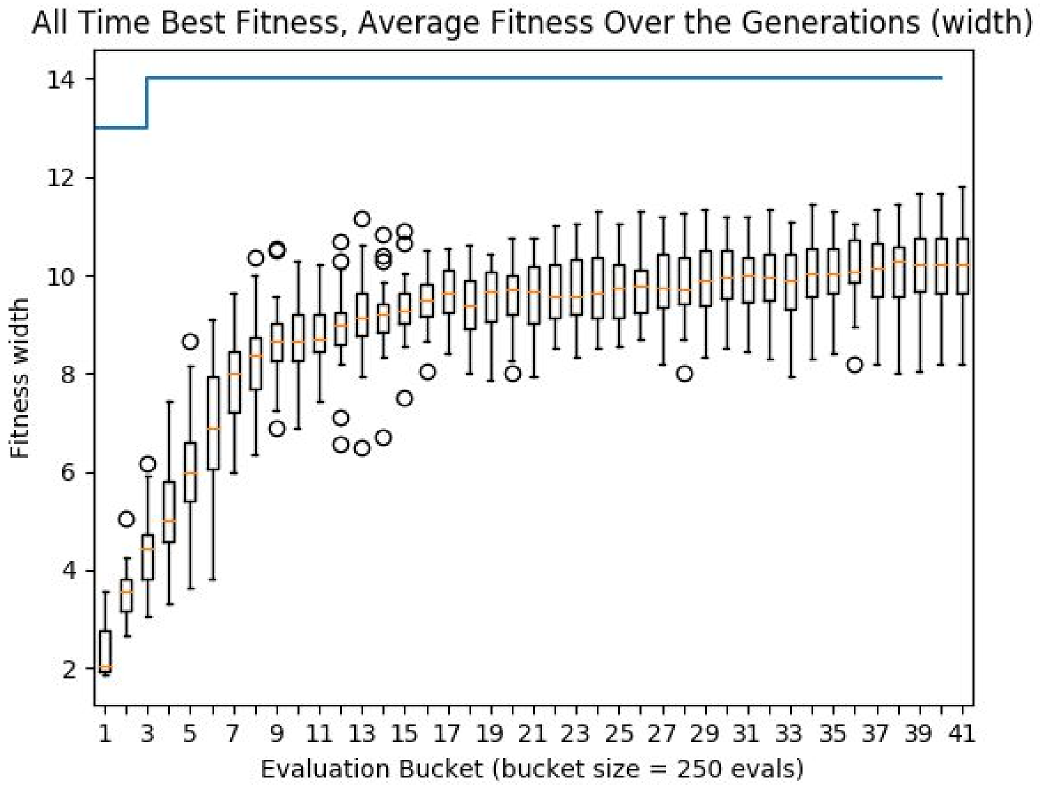
part 1

This is an exploration of the crowding strategy. Crowding showed interesting results and much improved results for width. Crowding seemed to better split the population between the different objectives allowing both to reach a stable convergence, unlike the results from general_1. However, these convergences seemed to stop abruptly and plateau early. However, this could be due to the small pool, 20, in the config.



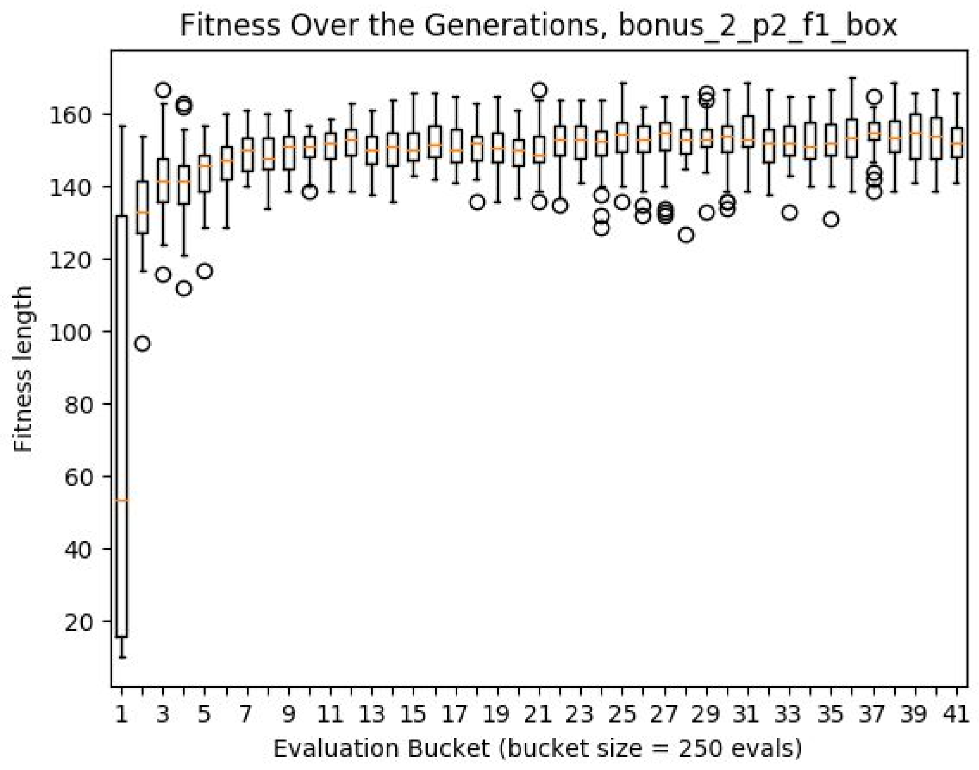

All Time Best Fitness, Average Fitness Over the Generations (length)
Fitness length
Evaluation Bucket (bucket size = 250 evals)
0
25
50
75
100
125
150
175
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41

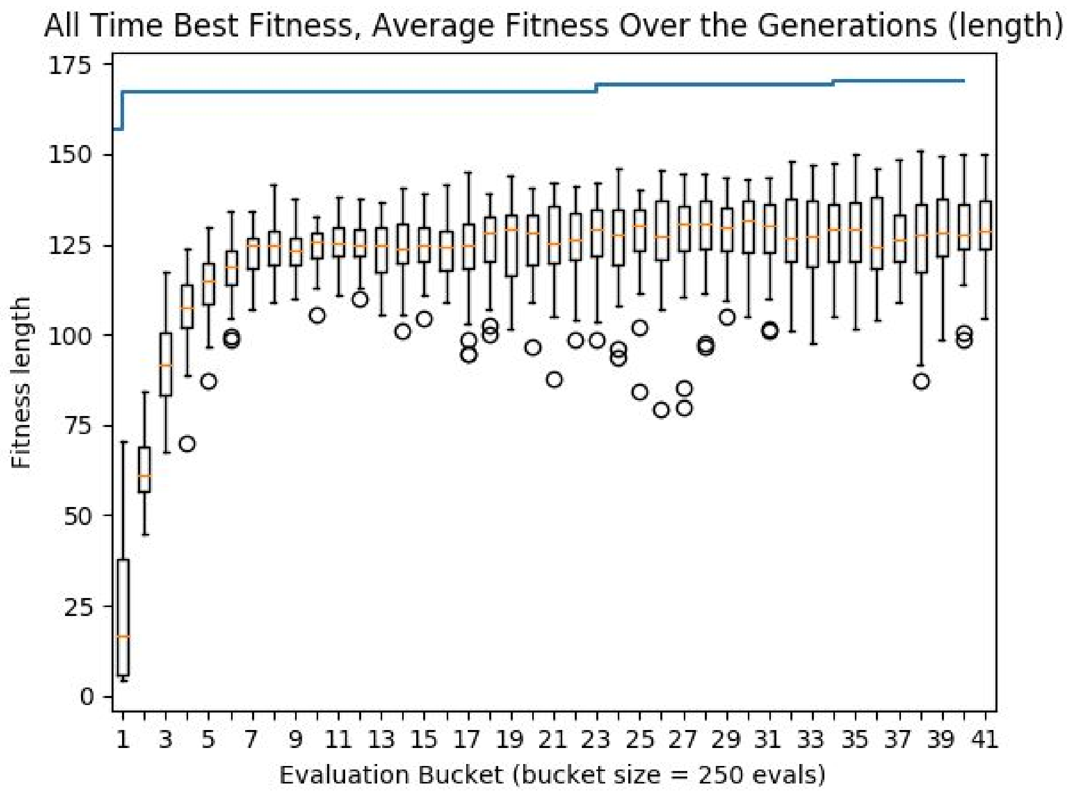

Fitness Over the Generations, bonus_2_p1_f1_box
Fitness width
Evaluation Bucket (bucket size = 250 evals)

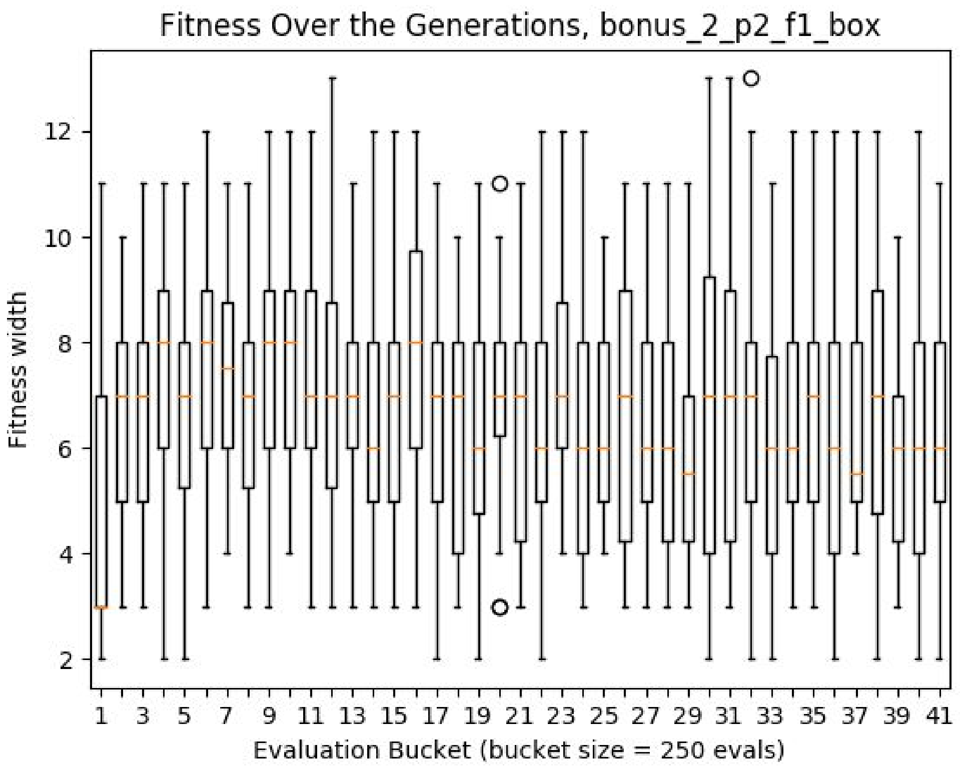

All Time Best Fitness, Average Fitness Over the Generations (width)
Fitness width
Evaluation Bucket (bucket size = 250 evals)

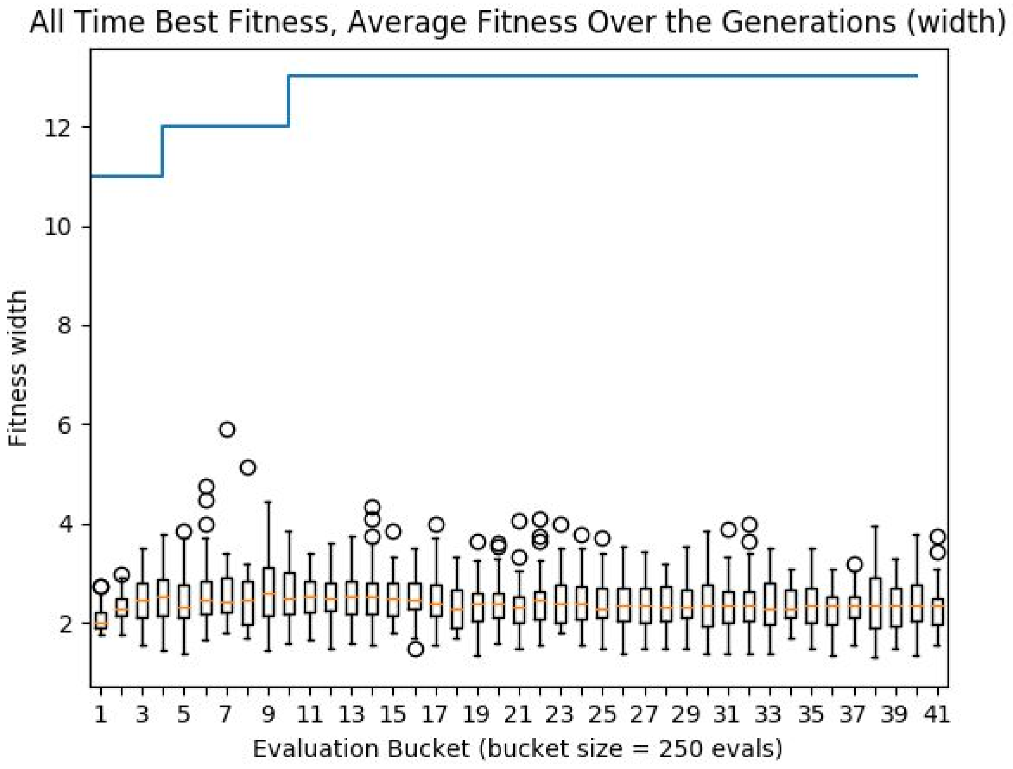
part 2

This is an exploration of the sharing strategy in comparison to the previous experiment and general_1. This sharing strategy seems to have negatives of both, but width does seem to perform slightly better than general_1. It seems general_1 is statistically better than both on length, and none are better on width - which the best seems to cap at 14 quite quickly.


Fitness Over the Generations, bonus_2_p2_f1_box
Fitness length
Evaluation Bucket (bucket size = 250 evals)
20
40
60
80
100
120
140
160
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41

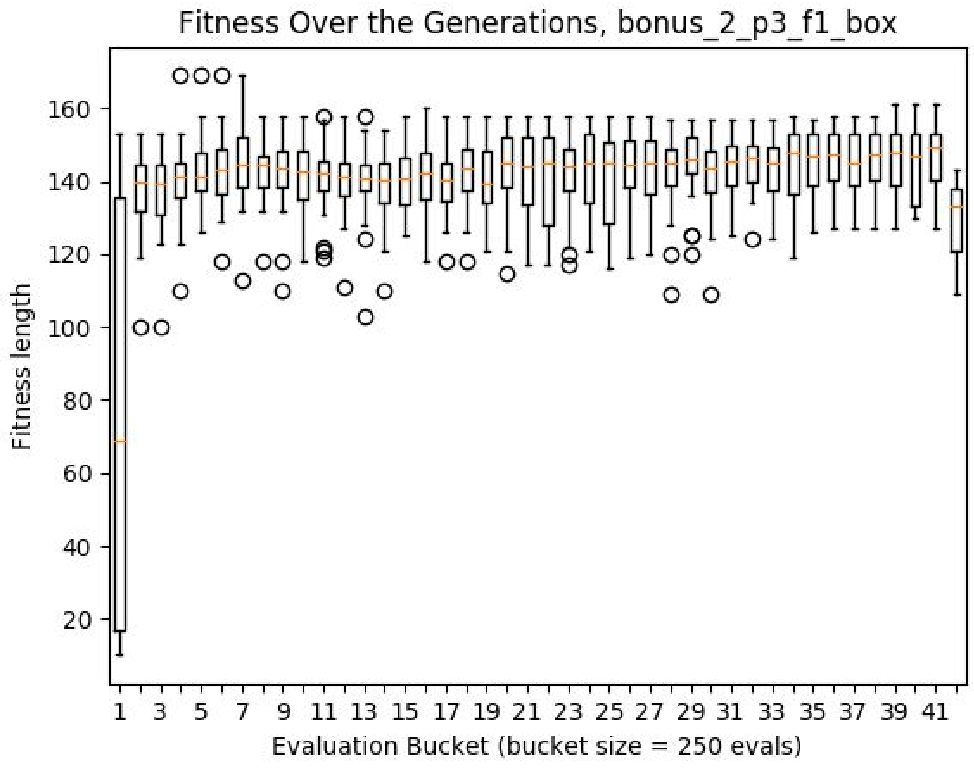

All Time Best Fitness, Average Fitness Over the Generations (length)
Fitness length
Evaluation Bucket (bucket size = 250 evals)
0
25
50
75
100
125
150
175
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41

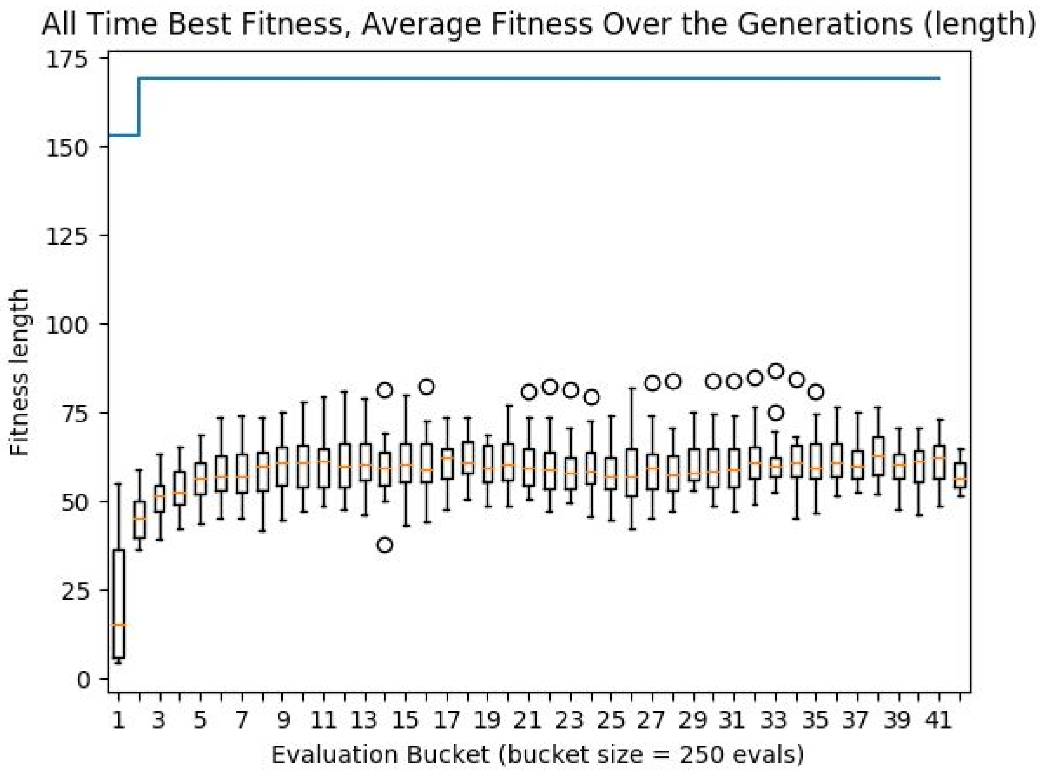

Fitness Over the Generations, bonus_2_p2_f1_box
Fitness width
Evaluation Bucket (bucket size = 250 evals)
2
4
6
8
10
12
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41

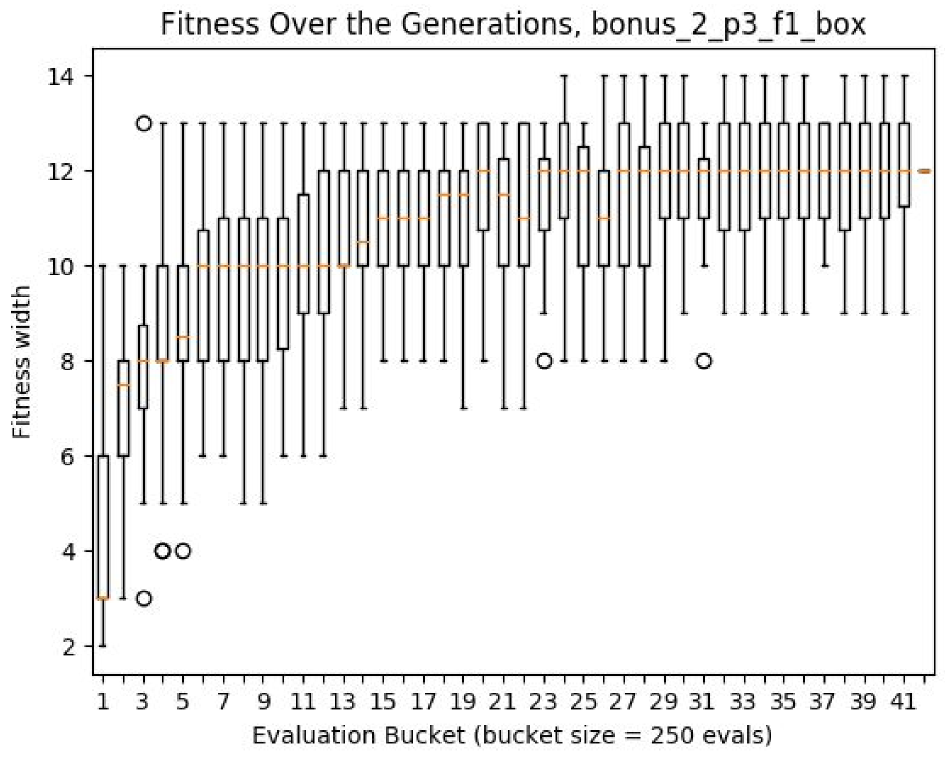

All Time Best Fitness, Average Fitness Over the Generations (width)
Fitness width
Evaluation Bucket (bucket size = 250 evals)
2
4
6
8
10
12
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41

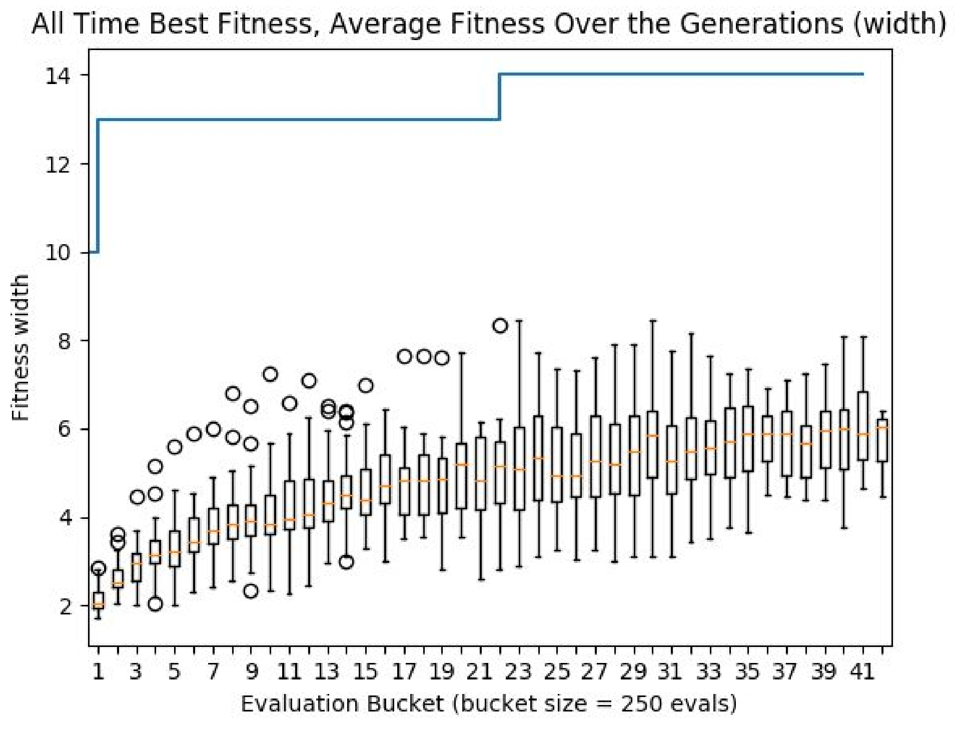
part 3

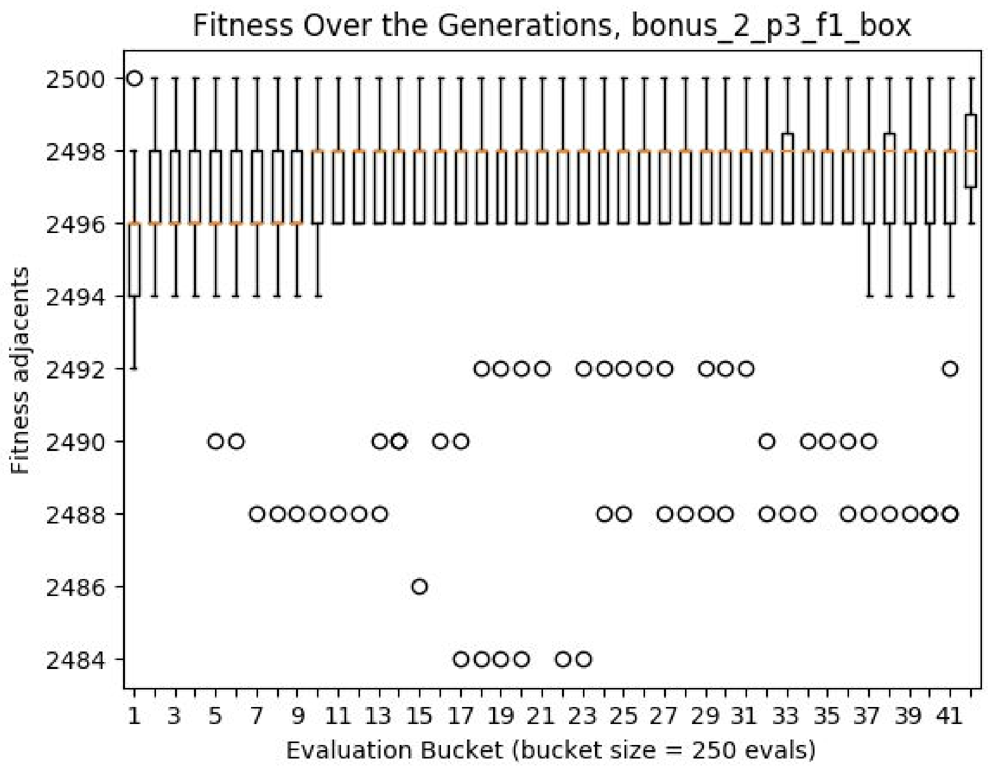
This is additional analysis of the adjacents objective function and how it interacts with length and width. While the length metric does worse, the width and adjacents fitness objectives do better than the bonus 1 counterpart. Sharing seems to enable better searching of all objective functions.

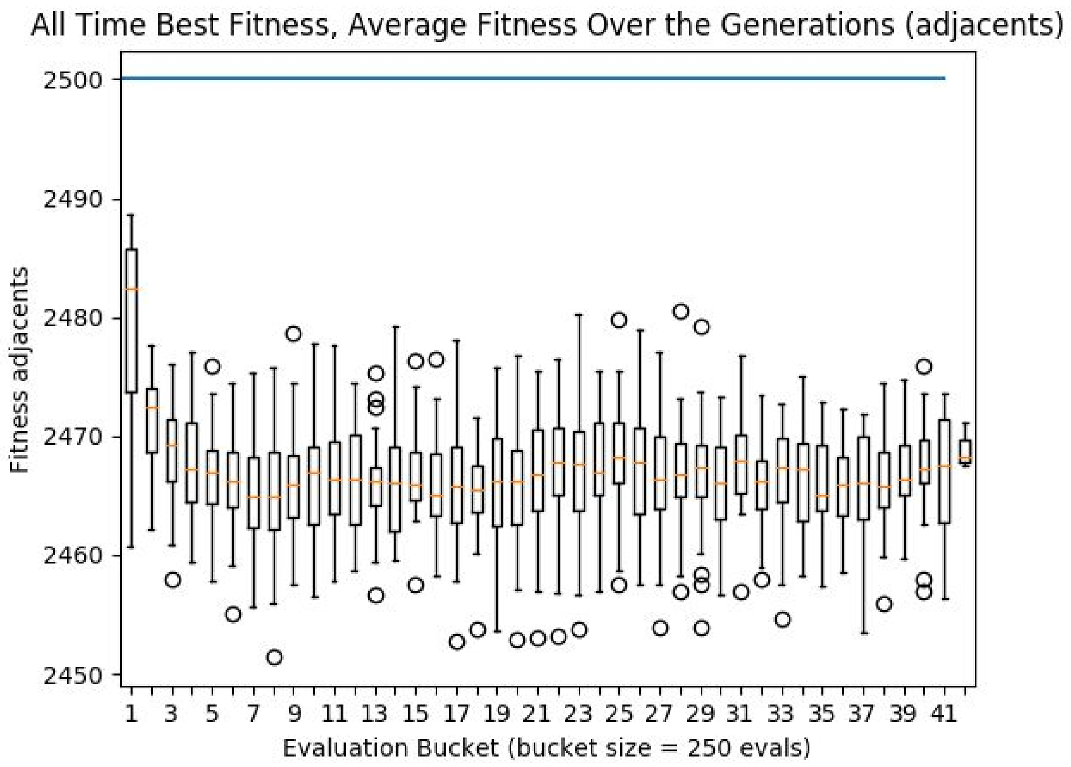





Fitness Over the Generations, bonus_2_p3_f1_box
Fitness width
Evaluation Bucket (bucket size = 250 evals)

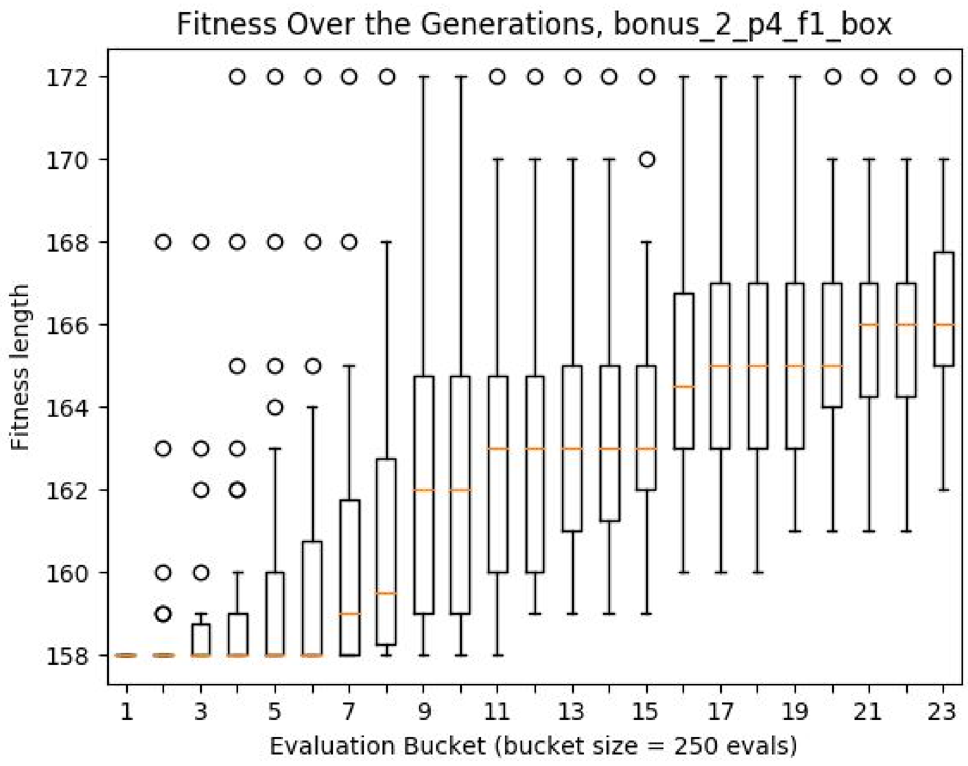

All Time Best Fitness, Average Fitness Over the Generations (width)
Fitness width
Evaluation Bucket (bucket size = 250 evals)

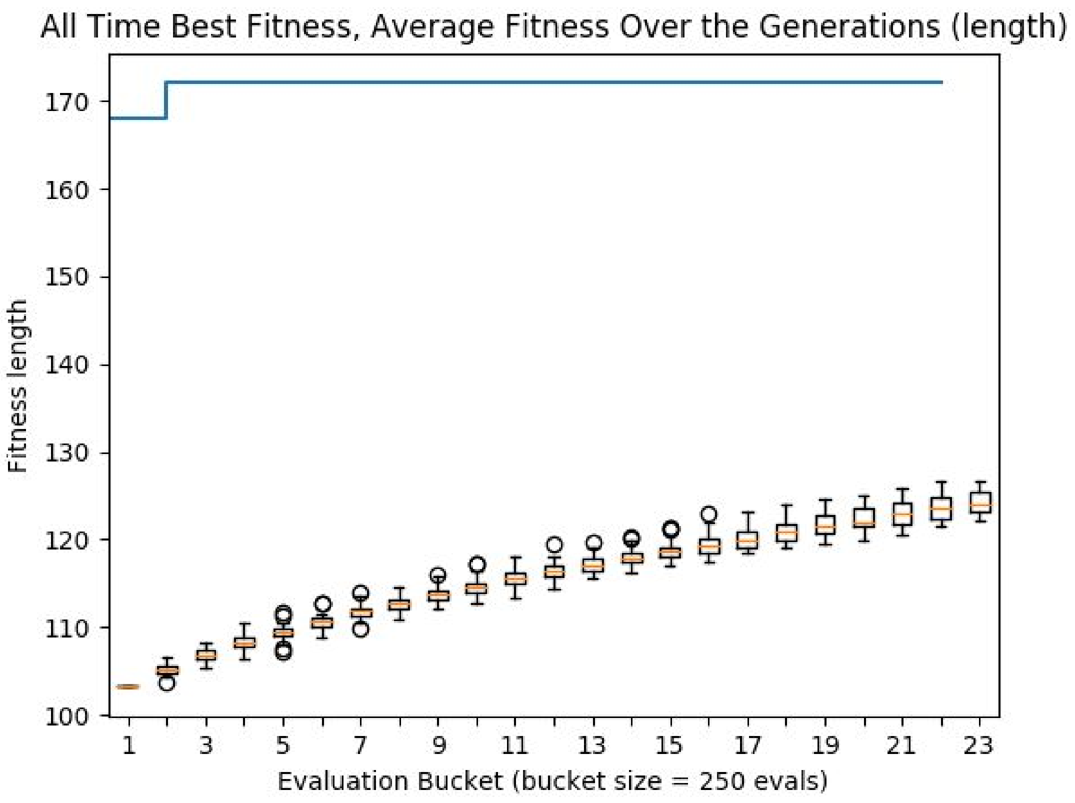

Fitness Over the Generations, bonus_2_p3_f1_box
Fitness adjacents
Evaluation Bucket (bucket size = 250 evals)


All Time Best Fitness, Average Fitness Over the Generations (adjacents)
Fitness adjacents
Evaluation Bucket (bucket size = 250 evals)

part 4

To explore crowding more, we compare it here to the expanded bonus 1 (part 2). This experiment builds off of the bonus part 1 data set as it did, and also focuses just on length, but it uses crowding to explore the space. This means that the diverse population of the triple objective solution file may be exploited more instead of out competing inferior solutions. It appears it might have been advantageous to extend the number of evals beyond 10,000, but we can see that the average fitness did not quickly skyrocket which enabled a more gradual search of the space. Statistically, crowding helped the search here and this can be considered better.

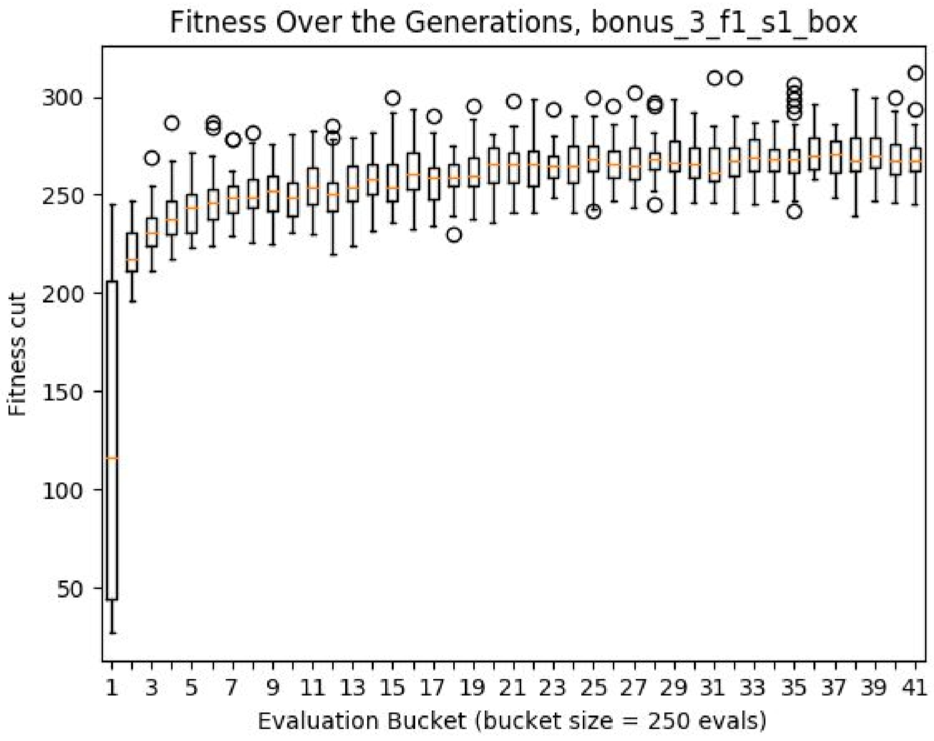


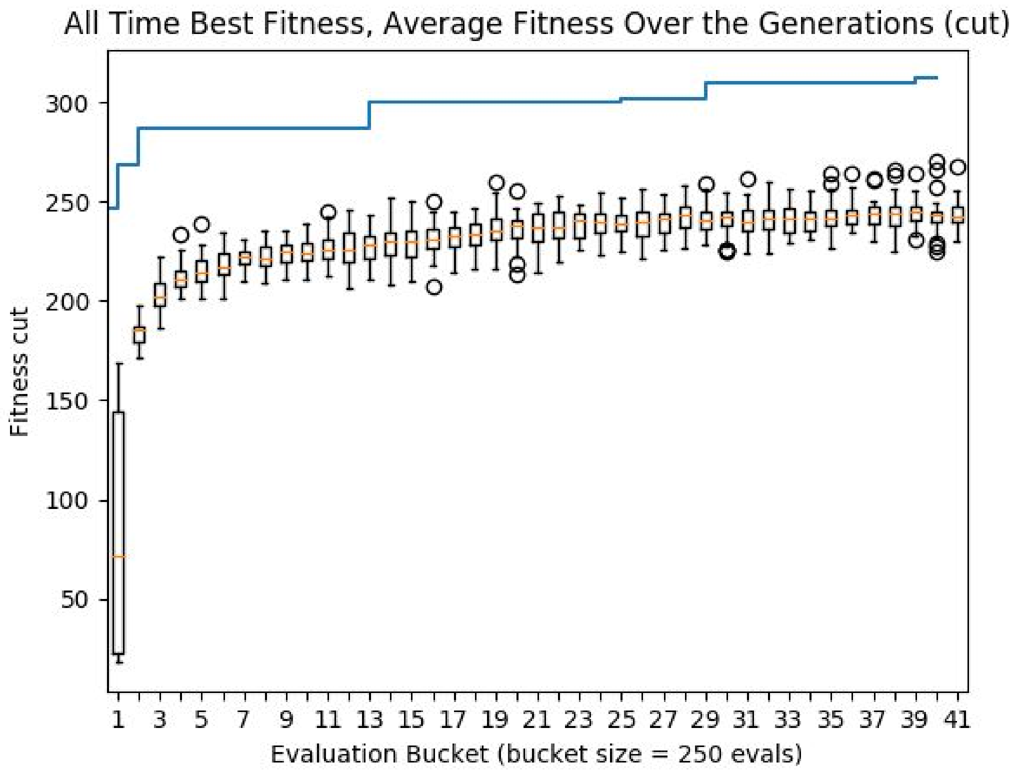

All Time Best Fitness, Average Fitness Over the Generations (length)
Fitness length
170
160
150
140
130
120
110
100
1
3
5
7
9
11
13
15
17
19
21
23
Evaluation Bucket (bucket size = 250 evals)

Bonus 3

step 1

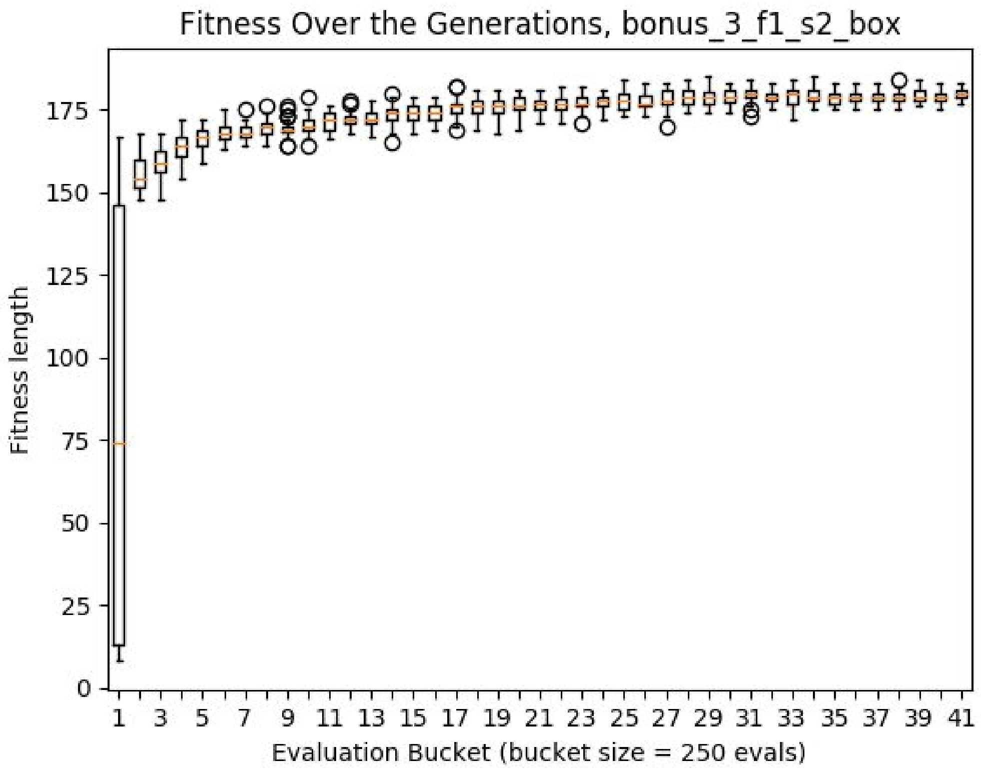
Here we explored a cut optimization objective. This means it is searching for ways to maximize adjacent edges of each shape. This experiment is just an initial experiment of the objective which we expand on after.

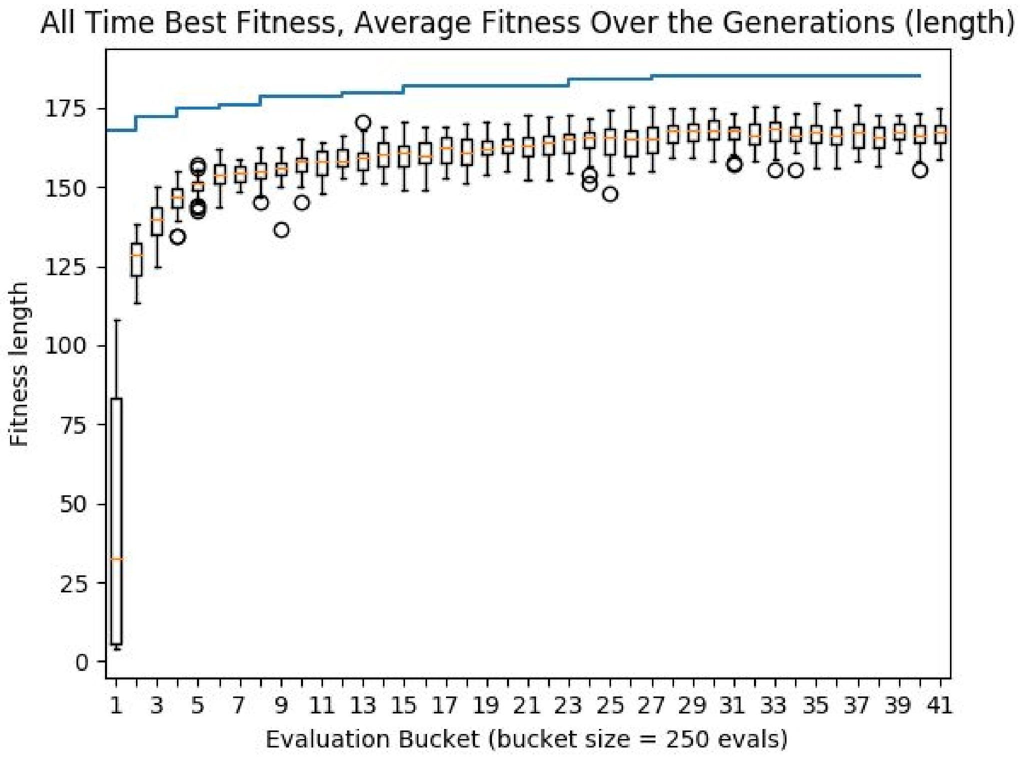


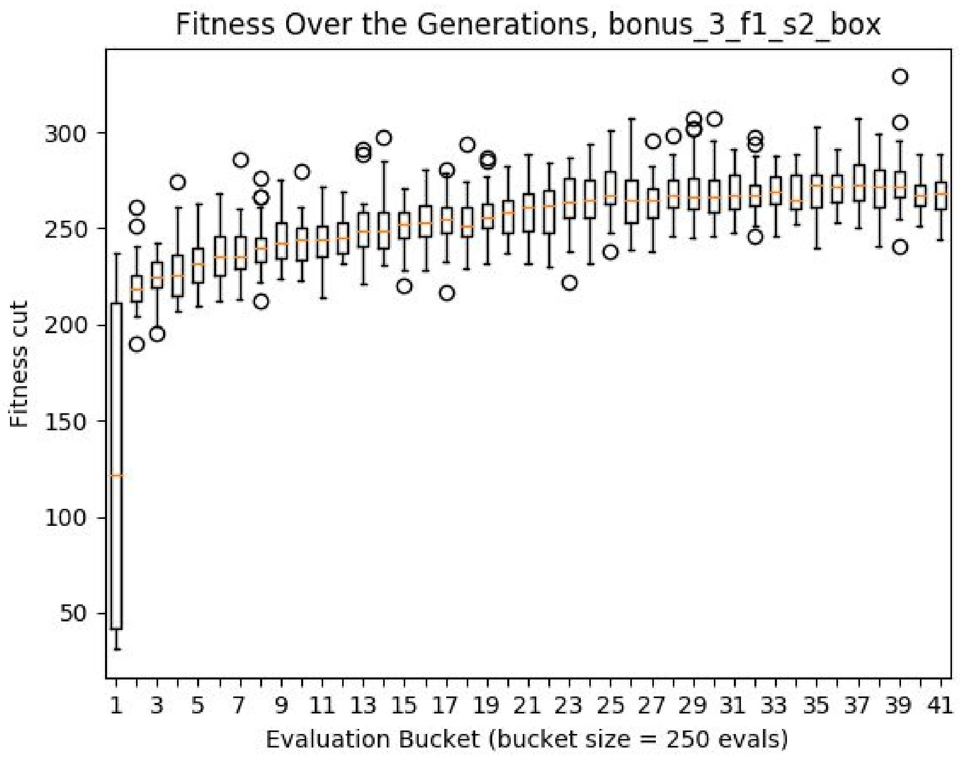

All Time Best Fitness, Average Fitness Over the Generations (cut)
Fitness cut
Evaluation Bucket (bucket size = 250 evals)
50
100
150
200
250
300
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41

step 2

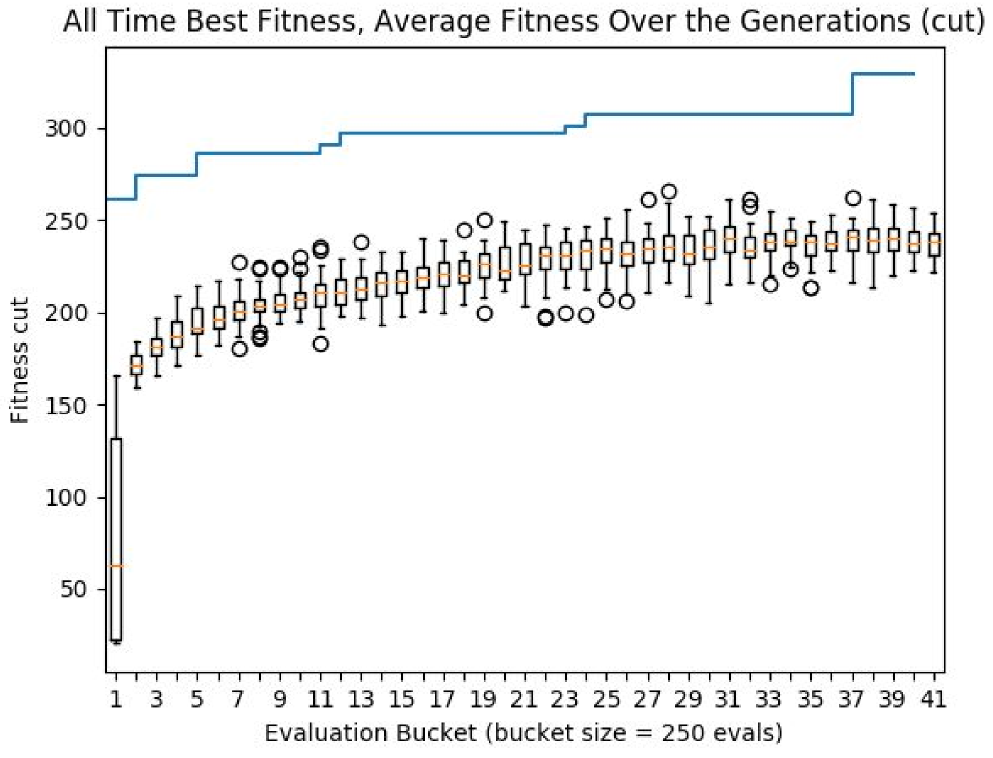
To further explore the metric, we do a multi objective search with length and cut. This alone finds statistically better results than general_1. This means, it appears these objectives do not conflict each other as much. Building off of this idea and previous ones, we use the solution output for this experiment in step 3. We also see the cut objective is barely hurt by the addition of the length objective.




All Time Best Fitness, Average Fitness Over the Generations (length)
Fitness length
Evaluation Bucket (bucket size = 250 evals)
0
25
50
75
100
125
150
175
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41



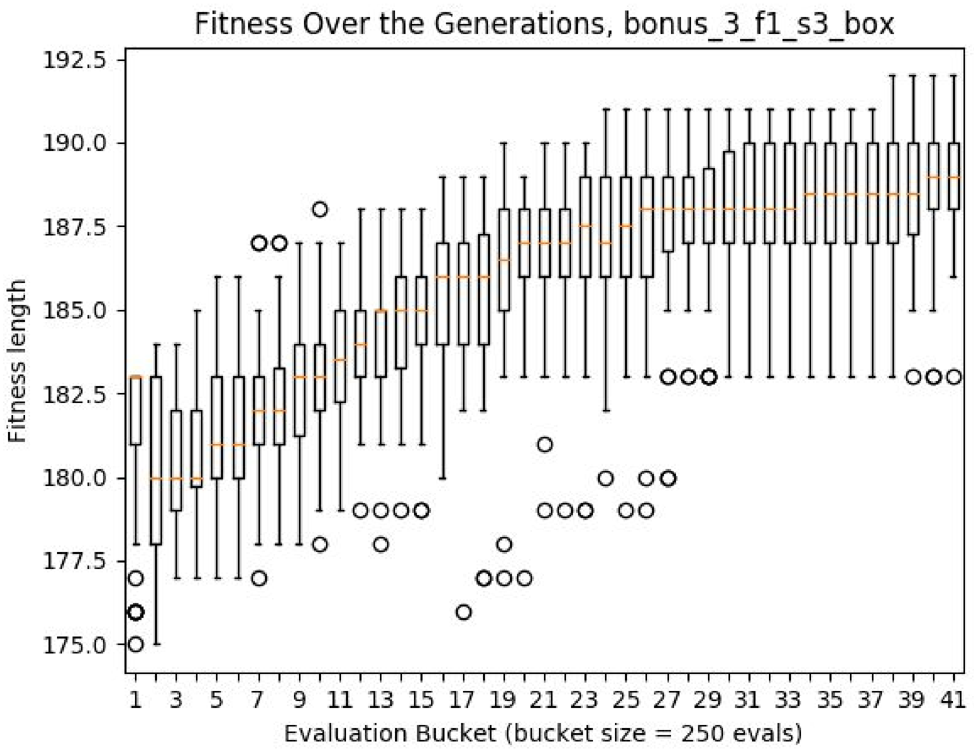
Fitness Over the Generations, bonus_3_f1_s2_box
Fitness cut
50
100
150
200
250
300
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
Evaluation Bucket (bucket size = 250 evals)

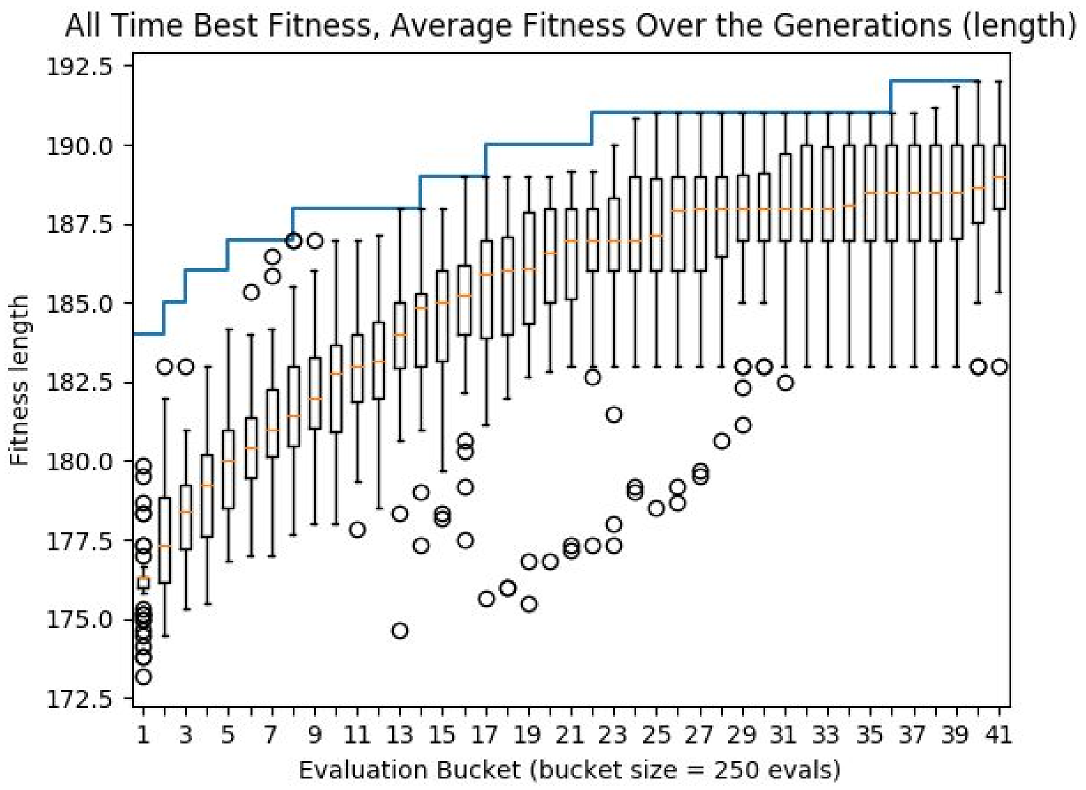


All Time Best Fitness, Average Fitness Over the Generations (cut)
Fitness cut
50
100
150
200
250
300
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
Evaluation Bucket (bucket size = 250 evals)

step 3

Using the solution output of step 2, we make another run with just the length objective. This initial search space enabled rapid discovery of highly performant solutions. This is by far the best statistical result for length fitness. This cut objective shows great effect in exploring different parts of the search space to enable a search to a good solution for our initial single objective. Perhaps unintuitively, we have shown that by adding an objective, we can increase the performance of our single objective.





All Time Best Fitness, Average Fitness Over the Generations (length)
Fitness length
Evaluation Bucket (bucket size = 250 evals)
192.5
190.0
187.5
185.0
182.5
180.0
177.5
175.0
172.5
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41