COMP SCI 5401 FS2017 Assignment 1d

Stuart Miller
sm67c@mst.edu

October 21, 2017

1 Overview

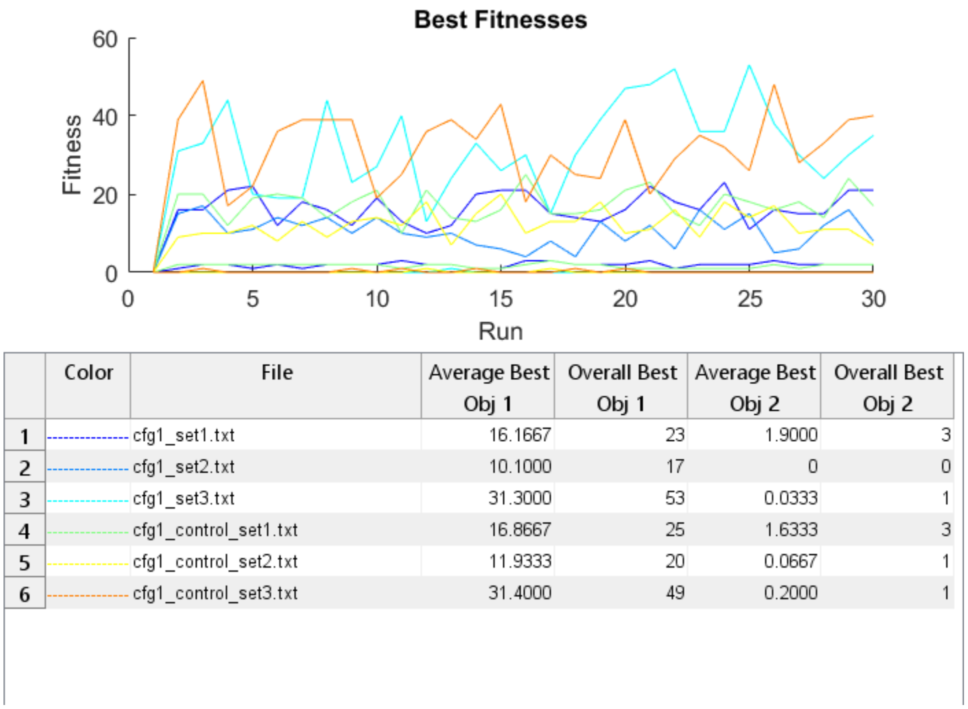
For assignment 1d, this iteration of the Stock-Cutting evolutionary algorithm introduces two new fitness measures: width and edge length. The main focus will be on length and width, while edge length will be discussed in Section 5 (Bonus 3). In order to facilitate this, a structure was created to hold various fitness measures and configuration options were added to point objective accessory to the configured fitness. The bulk of this assignment uses length as objective 1 and width as objective 2. For all of the fitness measures, minimized values were recorded by subtracting the observed measure from a set maximum value.

2 Configuration 1

For the first configuration, a highly selective EA will be employed. This EA uses k-tournaments with selection from a greedy 100% of the population. This is likely to allow the EA to get locked into a particular solution and not allow for innovation into possibly more optimal areas of the fitness landscape. A control group was also evaluated, where each EA had a k-tournament with selection from 50% of the population.

Figure 1 shows the best fitness for each objective. (Again, objective 1 is set to length and objective 2 is set to width.) The control group shows that it is able to usually generate better solutions, although not always. The greedy version of set 1 averaged a better best width, and the greedy versions of set 3 managed to find a single solution that was had a slightly better length than the control. In general though, it is clear that the greedy k-tournaments perform worse.

Figure 2 shows the points on the Pareto front. Note that the Pareto front is quite sparse. This is likely due to the specified way in which best Pareto fronts are determined. If a higher percentage of states in one Pareto front dominate the other than vice-versa, it is considered better. This gives an advantage to small Pareto fronts. For example a front with 3 solutions, 2 of which dominate is considered better than a front with 10 solutions, 6 of which dominate. Even though the 10 solution front has many more solutions, 3 solution front has a higher ratio and thus, wins.



	Color	File	Average Best Obj 1	Overall Best Obj 1	Average Best Obj 2	Overall Best Obj 2
1	┄┄┄	cfg1_set1.txt	16.1667	23	1.9000	3
2	┄┄┄	cfg1_set2.txt	10.1000	17	0	0
3	┄┄┄	cfg1_set3.txt	31.3000	53	0.0333	1
4	┄┄┄	cfg1_control_set1.txt	16.8667	25	1.6333	3
5	┄┄┄	cfg1_control_set2.txt	11.9333	20	0.0667	1
6	┄┄┄	cfg1_control_set3.txt	31.4000	49	0.2000	1

Figure 1: Configuration 1 Best Per-Objective Fitness

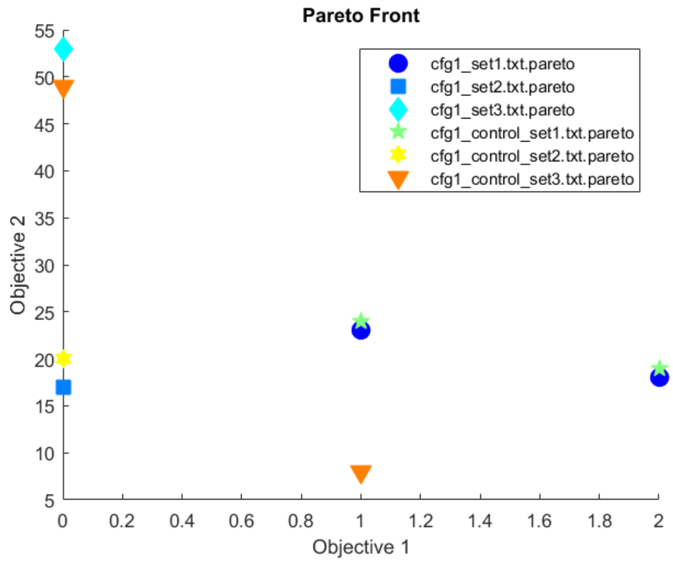


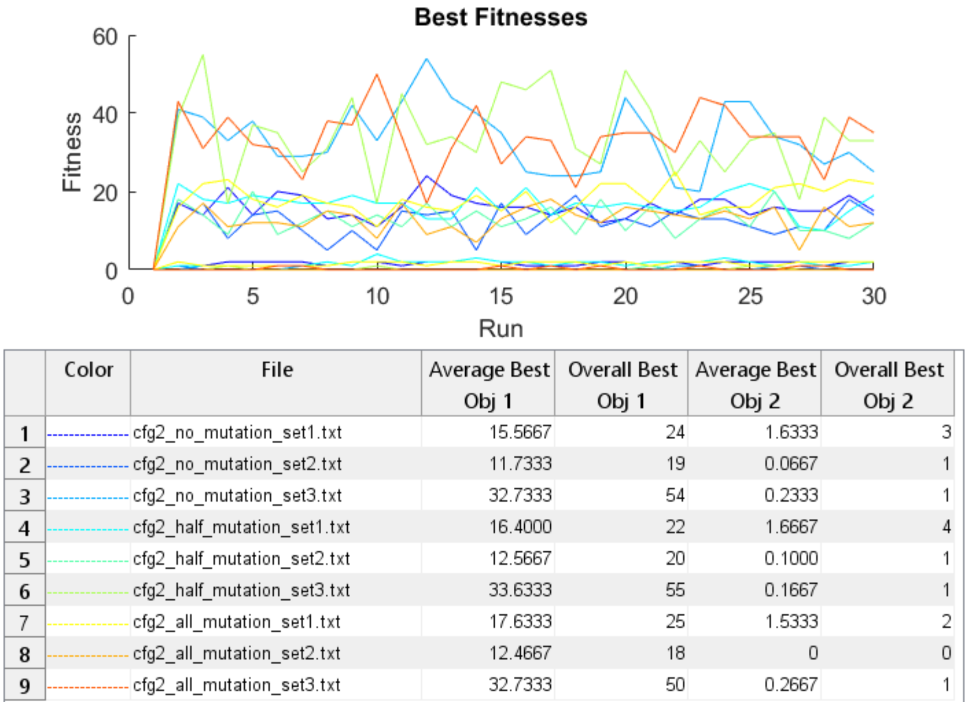
Figure 2: Configuration 1 Pareto Fronts

3 Configuration 2

The second configuration examines the effect of mutation rate. Three groups were used: one with no mutation at all, one with a 50% mutation chance, and one where every state was mutated once. Creep mutations were used, so each shape has a chance to change a single parameter up to 2 units in any direction. All other parameters were kept the same.

It is interesting that the fitness graph in Figure 3 show almost no difference in best fitness. The 50% mutation rate returns slightly better fitnesses, but not by much. This could be due to the fact the repair function takes care of creep on its own by moving the shapes a small amount. Adding a creep mutation seems to have little additional benefit. The benefit that is seen is likely due to the 50% mutation allowing for slightly increased exploration of the fitness landscape, while the 100% mutation provides to little direction to settle on definitively better solutions. Again, it is hard to see improvements in width due to the complexity of the data sets. Furthemore, smaller improvements are seen by the more complex data sets due to their increased complexity.

Due to the lack of diversity in widths, the Pareto fronts shown in Figure 4 tend to be small, showing points at the 0, 1, and 2 width measures.



	Color	File	Average Best Obj 1	Overall Best Obj 1	Average Best Obj 2	Overall Best Obj 2
1	----	cfg2_no_mutation_set1.txt	15.5667	24	1.6333	3
2	----	cfg2_no_mutation_set2.txt	11.7333	19	0.0667	1
3	----	cfg2_no_mutation_set3.txt	32.7333	54	0.2333	1
4	----	cfg2_half_mutation_set1.txt	16.4000	22	1.6667	4
5	----	cfg2_half_mutation_set2.txt	12.5667	20	0.1000	1
6	----	cfg2_half_mutation_set3.txt	33.6333	55	0.1667	1
7	----	cfg2_all_mutation_set1.txt	17.6333	25	1.5333	2
8	----	cfg2_all_mutation_set2.txt	12.4667	18	0	0
9	----	cfg2_all_mutation_set3.txt	32.7333	50	0.2667	1

Figure 3: Configuration 2 Best Per-Objective Fitness

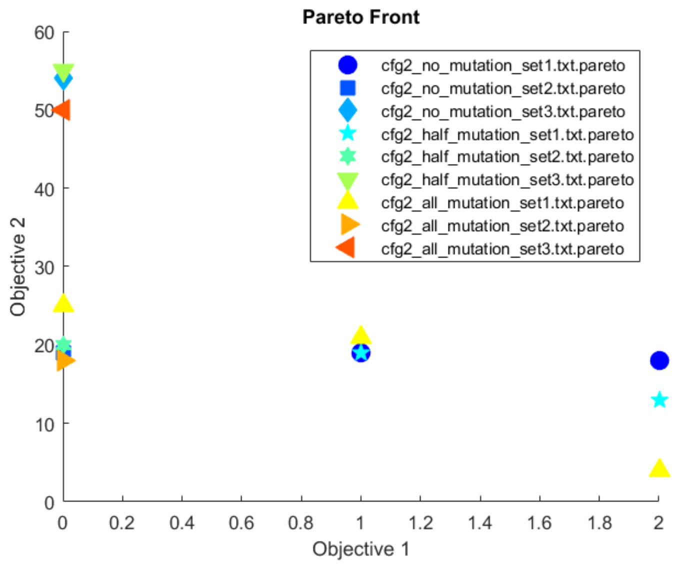


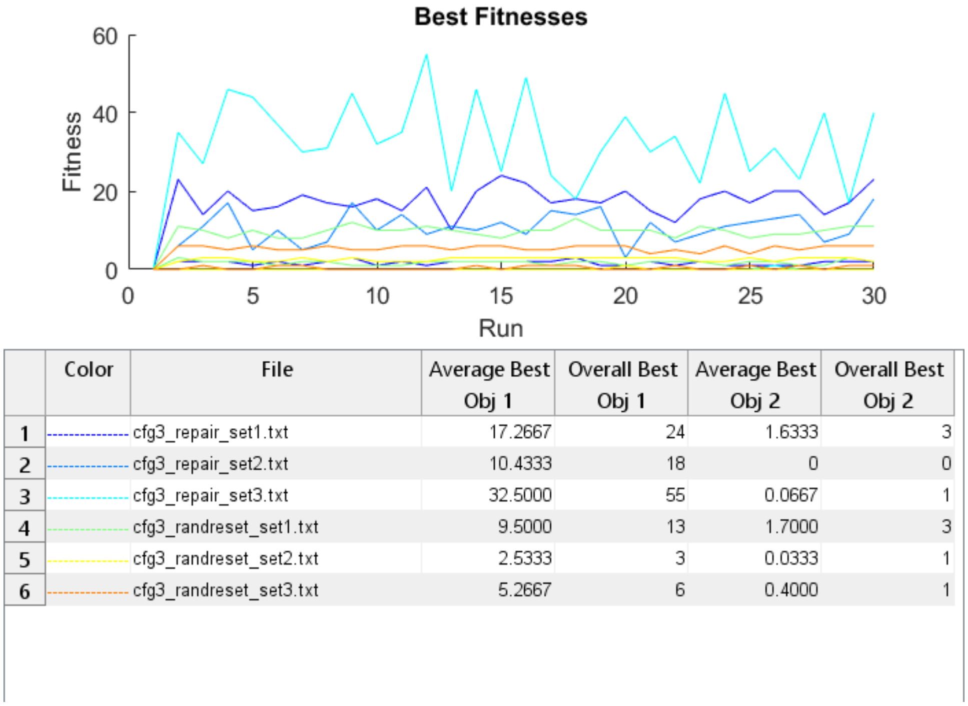
Figure 4: Configuration 2 Pareto Fronts

4 Configuration 3

Finally, the third configuration toggles the repair function. Calling the repair function was compared to using random resets for invalid shape placements.

This configuration displays undoubtedly the largest divide yet. Figure 5 shows a clear difference between the fitnesses from repair constraint resolution verses the random reset constraint resolution. This divide is a little less clear in the second objective (width), although this is in line with the difficulty to establish any significant improvements in width across the other configurations run above. As is to be expected, the more complex data sets show lesser improvements with set 2 having a very low improvement.

Again, the Pareto fronts shown in Figure 6 have few members, showing peaks at the 0 and 1 widths and a few at 2.



	Color	File	Average Best Obj 1	Overall Best Obj 1	Average Best Obj 2	Overall Best Obj 2
1	--------	cfg3_repair_set1.txt	17.2667	24	1.6333	3
2	--------	cfg3_repair_set2.txt	10.4333	18	0	0
3	--------	cfg3_repair_set3.txt	32.5000	55	0.0667	1
4	--------	cfg3_randreset_set1.txt	9.5000	13	1.7000	3
5	--------	cfg3_randreset_set2.txt	2.5333	3	0.0333	1
6	--------	cfg3_randreset_set3.txt	5.2667	6	0.4000	1

Figure 5: Configuration 3 Best Per-Objective Fitness

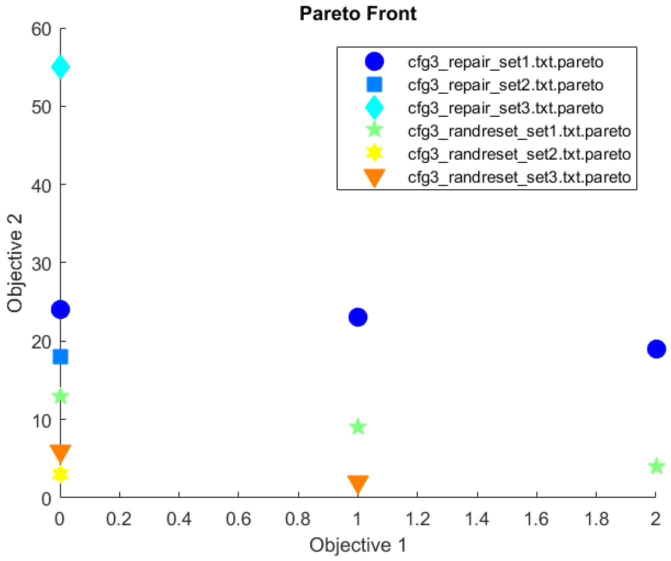


Figure 6: Configuration 3 Pareto Fronts

5 Bonus 3