Fixed convergence experiment – take a fixed graph and run evolution several times to see how the convergence varies

If convergence varies, compare performance of different strategies

Gen. 0 (0.00%): Max/Min/Avg Fitness(Raw) [6.07(10.00)/4.02(0.00)/5.06(5.06)]

Gen. 10 (50.00%): Max/Min/Avg Fitness(Raw) [7.59(10.00)/4.84(2.00)/6.33(6.33)]

Gen. 20 (100.00%): Max/Min/Avg Fitness(Raw) [8.23(10.00)/5.17(3.00)/6.86(6.86)]

Total time elapsed: 1022.681 seconds.

[['1', '3', '1', '4', '3'], ['3', '2', '2', '1', '4'], ['3', '2', '2', '1', '4'], ['3', '2', '2', '1', '4'], ['3', '2', '2', '1', '4'], ['3', '2', '2', '1', '4'], ['2', '2', '3', '1', '4'], ['2', '2', '3', '1', '4'], ['2', '2', '3', '1', '4'], ['2', '2', '3', '1', '4'], ['2', '2', '3', '1', '4'], ['2', '2', '3', '1', '4'], ['2', '2', '3', '1', '4'], ['2', '2', '3', '1', '4'], ['2', '2', '3', '1', '4'], ['2', '2', '3', '1', '4'], ['2', '2', '3', '1', '4'], ['2', '2', '3', '1', '4'], ['2', '2', '3', '1', '4'], ['2', '2', '3', '1', '4'], ['1', '2', '3', '2', '4']]

Gen. 0 (0.00%): Max/Min/Avg Fitness(Raw) [5.40(8.00)/3.60(1.00)/4.50(4.50)]

Gen. 10 (50.00%): Max/Min/Avg Fitness(Raw) [8.61(10.00)/5.05(3.00)/7.18(7.17)]

Gen. 20 (100.00%): Max/Min/Avg Fitness(Raw) [9.08(10.00)/5.97(5.00)/7.56(7.56)]

Total time elapsed: 1651.161 seconds.

[['0', '1', '0', '3', '1'], ['0', '2', '0', '0', '4'], ['0', '2', '0', '0', '4'], ['0', '2', '0', '0', '4'], ['0', '2', '0', '0', '4'], ['0', '2', '0', '0', '4'], ['0', '2', '0', '0', '4'], ['0', '2', '0', '0', '4'], ['0', '2', '0', '0', '4'], ['0', '2', '0', '0', '4'], ['0', '2', '0', '0', '4'], ['0', '2', '0', '0', '4'], ['0', '2', '0', '0', '4'], ['0', '0', '0', '2', '4'], ['0', '0', '0', '2', '4'], ['0', '0', '0', '2', '4'], ['0', '0', '0', '2', '4'], ['0', '2', '0', '2', '4'], ['0', '2', '0', '2', '4'], ['0', '0', '0', '2', '4'], ['0', '0', '0', '2', '4']]

Gen. 0 (0.00%): Max/Min/Avg Fitness(Raw) [5.46(9.00)/3.62(0.00)/4.55(4.55)]

Gen. 10 (50.00%): Max/Min/Avg Fitness(Raw) [9.00(10.00)/5.40(4.00)/7.50(7.50)]

Gen. 20 (100.00%): Max/Min/Avg Fitness(Raw) [9.09(10.00)/5.34(4.00)/7.58(7.58)]

Total time elapsed: 465.857 seconds.

[['0', '4', '4', '0', '4'], ['0', '3', '2', '2', '4'], ['0', '3', '2', '2', '4'], ['0', '3', '2', '2', '4'], ['0', '3', '2', '2', '4'], ['0', '3', '2', '2', '4'], ['0', '3', '2', '2', '4'], ['0', '3', '2', '2', '4'], ['0', '3', '2', '2', '4'], ['0', '3', '2', '2', '4'], ['0', '3', '2', '2', '4'], ['0', '3', '2', '2', '4'], ['0', '3', '2', '2', '4'], ['0', '3', '2', '2', '4'], ['0', '3', '2', '2', '4'], ['0', '3', '2', '2', '4'], ['0', '3', '2', '2', '4'], ['0', '3', '2', '2', '4'], ['0', '3', '2', '2', '4'], ['2', '3', '0', '2', '4'], ['0', '3', '2', '2', '4']]

Gen. 0 (0.00%): Max/Min/Avg Fitness(Raw) [5.67(9.00)/3.68(0.00)/4.73(4.72)]

Gen. 10 (50.00%): Max/Min/Avg Fitness(Raw) [7.70(10.00)/5.19(3.00)/6.41(6.41)]

Gen. 20 (100.00%): Max/Min/Avg Fitness(Raw) [7.16(10.00)/4.79(2.00)/5.96(5.96)]

Total time elapsed: 1100.251 seconds.

[['3', '4', '2', '3', '2'], ['3', '0', '2', '0', '2'], ['3', '0', '2', '0', '2'], ['3', '0', '2', '0', '0'], ['3', '0', '2', '0', '0'], ['3', '0', '2', '0', '2'], ['3', '0', '2', '0', '2'], ['3', '0', '2', '0', '2'], ['3', '0', '2', '0', '2'], ['3', '0', '2', '0', '2'], ['3', '0', '2', '0', '2'], ['3', '0', '2', '0', '2'], ['3', '0', '2', '0', '2'], ['3', '0', '2', '0', '2'], ['3', '0', '2', '0', '2'], ['3', '0', '2', '0', '2'], ['3', '0', '2', '0', '2'], ['3', '0', '2', '0', '2'], ['3', '0', '2', '0', '2'], ['3', '0', '2', '0', '2'], ['3', '0', '2', '0', '2']]

Gen. 0 (0.00%): Max/Min/Avg Fitness(Raw) [6.39(10.00)/4.34(1.00)/5.33(5.33)]

Gen. 10 (50.00%): Max/Min/Avg Fitness(Raw) [8.50(10.00)/4.61(2.00)/7.09(7.09)]

Gen. 20 (100.00%): Max/Min/Avg Fitness(Raw) [9.09(10.00)/5.34(4.00)/7.58(7.58)]

Total time elapsed: 1490.197 seconds.

[['2', '4', '0', '1', '3'], ['2', '0', '0', '0', '4'], ['3', '0', '2', '0', '4'], ['2', '0', '0', '0', '4'], ['2', '0', '0', '0', '4'], ['2', '0', '0', '0', '4'], ['2', '0', '0', '0', '4'], ['2', '0', '0', '0', '4'], ['2', '0', '0', '0', '4'], ['2', '0', '0', '0', '4'], ['2', '0', '0', '0', '4'], ['2', '0', '0', '0', '4'], ['2', '0', '0', '0', '4'], ['0', '0', '2', '0', '4'], ['0', '0', '2', '0', '4'], ['0', '0', '2', '0', '4'], ['0', '0', '2', '0', '4'], ['0', '0', '2', '0', '4'], ['0', '0', '2', '0', '4'], ['0', '0', '2', '0', '4'], ['0', '0', '2', '0', '4']]

Gen. 0 (0.00%): Max/Min/Avg Fitness(Raw) [5.58(9.00)/3.66(0.00)/4.65(4.65)]

Gen. 10 (50.00%): Max/Min/Avg Fitness(Raw) [8.05(10.00)/5.20(3.00)/6.71(6.71)]

Gen. 20 (100.00%): Max/Min/Avg Fitness(Raw) [8.28(10.00)/5.16(3.00)/6.90(6.90)]

Total time elapsed: 606.201 seconds.

[['3', '4', '2', '3', '0'], ['1', '3', '4', '2', '4'], ['0', '3', '3', '2', '2'], ['0', '3', '3', '2', '2'], ['1', '3', '3', '2', '4'], ['0', '3', '3', '2', '2'], ['0', '3', '3', '2', '2'], ['0', '3', '3', '2', '2'], ['0', '3', '3', '2', '2'], ['0', '3', '3', '2', '2'], ['0', '3', '3', '2', '2'], ['0', '3', '3', '2', '2'], ['0', '3', '3', '2', '2'], ['0', '3', '3', '2', '2'], ['0', '3', '3', '2', '2'], ['0', '3', '3', '2', '2'], ['0', '3', '3', '2', '2'], ['0', '3', '3', '2', '2'], ['0', '3', '3', '2', '2'], ['0', '3', '3', '2', '2'], ['2', '3', '3', '2', '0']]

RESULTS

Six trials on this graph

['1', '2', '3', '2', '4']

['0', '0', '0', '2', '4']

['0', '3', '2', '2', '4']

['3', '0', '2', '0', '2']

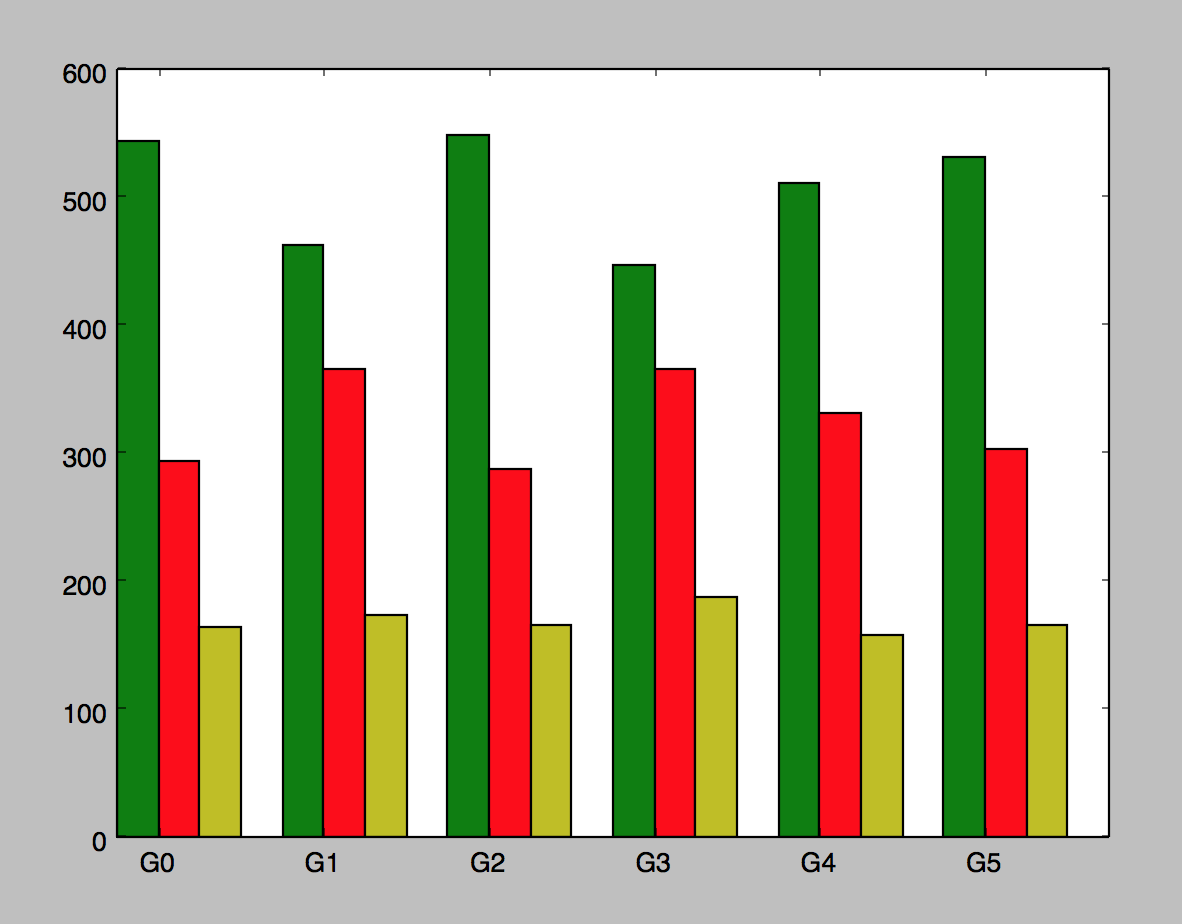
['0', '0', '2', '0', '4']

[‘2’, '3', '3', '2', '0']

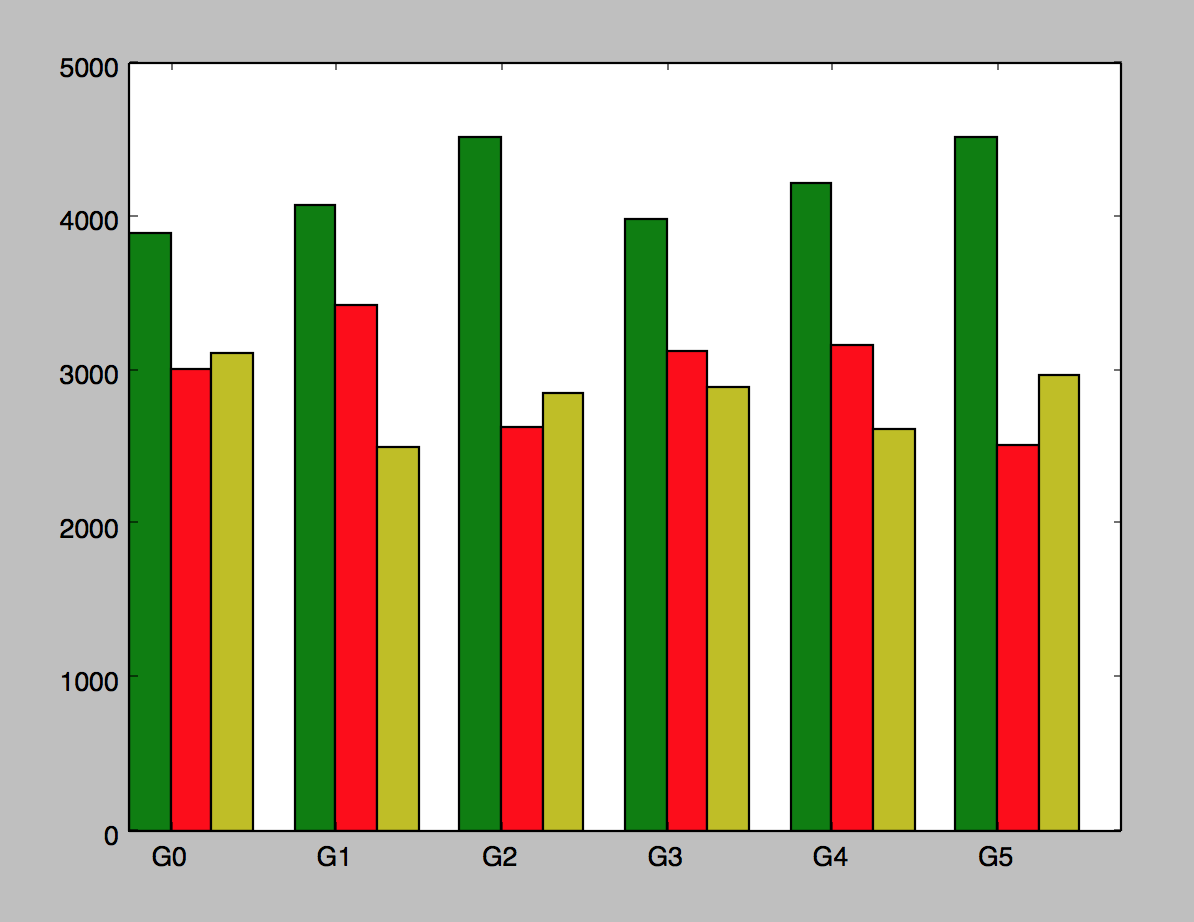
[['1', '2', '3', '2', '4'],['0', '0', '0', '2', '4'],['0', '3', '2', '2', '4'],['3', '0', '2', '0', '2'],['0', '0', '2', '0', '4'],['2', '3', '3', '2', '0']]

GRAPHS

Games = 1000



games = 10000



Conclusion:

Not one clearly better than the other

Local minima that perform about the same

Next: is the chromosome not mutating enough to get out of closest local minima and into maxima

Currently:

>two point crossover

>swap mutation

Try:

>different mutation type

genome.mutator.set(Mutators.G1DListMutatorIntegerRange)

Gen. 0 (0.00%): Max/Min/Avg Fitness(Raw) [5.14(9.00)/3.51(0.00)/4.29(4.29)]

Gen. 10 (50.00%): Max/Min/Avg Fitness(Raw) [6.84(9.00)/4.08(1.00)/5.70(5.70)]

Gen. 20 (100.00%): Max/Min/Avg Fitness(Raw) [6.75(9.00)/4.75(3.00)/5.63(5.62)]

Total time elapsed: 69.697 seconds.

[['0', '4', '4', '1', '2'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4']]

Gen. 0 (0.00%): Max/Min/Avg Fitness(Raw) [5.62(9.00)/3.89(1.00)/4.69(4.69)]

Gen. 10 (50.00%): Max/Min/Avg Fitness(Raw) [9.12(10.00)/4.69(3.00)/7.60(7.60)]

Gen. 20 (100.00%): Max/Min/Avg Fitness(Raw) [9.39(10.00)/5.79(5.00)/7.82(7.83)]

Total time elapsed: 221.981 seconds.

[['0', '4', '2', '3', '0'], ['2', '3', '0', '4', '3'], ['1', '1', '3', '0', '4'], ['1', '1', '3', '0', '4'], ['3', '1', '3', '0', '4'], ['3', '1', '3', '0', '4'], ['3', '2', '3', '0', '4'], ['3', '1', '3', '0', '4'], ['3', '1', '3', '0', '4'], ['3', '2', '3', '2', '4'], ['3', '2', '3', '2', '4'], ['3', '2', '3', '0', '4'], ['3', '2', '3', '0', '4'], ['3', '2', '3', '2', '4'], ['3', '2', '3', '2', '4'], ['3', '2', '3', '2', '4'], ['3', '2', '3', '2', '4'], ['3', '2', '3', '2', '4'], ['3', '2', '3', '2', '4'], ['3', '2', '3', '2', '4'], ['3', '2', '3', '2', '4']]

iteration: 1

Gen. 0 (0.00%): Max/Min/Avg Fitness(Raw) [5.59(9.00)/3.66(0.00)/4.66(4.66)]

Gen. 10 (50.00%): Max/Min/Avg Fitness(Raw) [8.12(10.00)/5.19(3.00)/6.76(6.76)]

Gen. 20 (100.00%): Max/Min/Avg Fitness(Raw) [7.83(10.00)/5.20(3.00)/6.52(6.53)]

Total time elapsed: 535.879 seconds.

[['0', '3', '0', '0', '0'], ['0', '0', '3', '3', '4'], ['0', '0', '3', '3', '4'], ['0', '0', '3', '3', '4'], ['0', '0', '0', '3', '4'], ['0', '0', '0', '3', '4'], ['0', '0', '0', '3', '4'], ['0', '0', '0', '3', '4'], ['0', '0', '0', '3', '4'], ['0', '0', '0', '3', '4'], ['0', '0', '0', '3', '4'], ['0', '0', '0', '3', '4'], ['0', '2', '0', '3', '4'], ['0', '0', '0', '3', '4'], ['0', '0', '0', '3', '4'], ['0', '0', '0', '3', '4'], ['0', '0', '0', '3', '4'], ['0', '0', '0', '3', '4'], ['0', '0', '0', '3', '4'], ['0', '0', '0', '3', '4'], ['0', '0', '0', '3', '4']]

iteration: 2

Gen. 0 (0.00%): Max/Min/Avg Fitness(Raw) [5.80(10.00)/3.93(0.00)/4.84(4.84)]

Gen. 10 (50.00%): Max/Min/Avg Fitness(Raw) [7.31(10.00)/4.82(2.00)/6.09(6.09)]

Gen. 20 (100.00%): Max/Min/Avg Fitness(Raw) [7.67(10.00)/5.19(3.00)/6.39(6.39)]

Total time elapsed: 551.571 seconds.

[['4', '4', '2', '1', '3'], ['3', '0', '3', '3', '0'], ['3', '0', '3', '3', '0'], ['3', '0', '3', '3', '0'], ['3', '0', '3', '3', '0'], ['3', '0', '3', '3', '0'], ['3', '0', '3', '3', '0'], ['3', '0', '3', '3', '0'], ['3', '0', '3', '3', '0'], ['3', '0', '3', '3', '0'], ['2', '1', '3', '3', '0'], ['2', '1', '3', '3', '0'], ['2', '1', '3', '3', '0'], ['2', '1', '3', '3', '0'], ['2', '1', '3', '3', '0'], ['2', '1', '3', '3', '0'], ['2', '1', '3', '3', '0'], ['2', '1', '3', '3', '0'], ['2', '1', '3', '3', '0'], ['2', '1', '3', '3', '0'], ['2', '1', '3', '3', '0']]

iteration: 3

Gen. 0 (0.00%): Max/Min/Avg Fitness(Raw) [5.34(9.00)/3.58(0.00)/4.45(4.45)]

Gen. 10 (50.00%): Max/Min/Avg Fitness(Raw) [8.62(10.00)/5.05(3.00)/7.19(7.19)]

Gen. 20 (100.00%): Max/Min/Avg Fitness(Raw) [8.64(10.00)/5.55(4.00)/7.20(7.20)]

Total time elapsed: 483.064 seconds.

[['1', '2', '1', '0', '1'], ['1', '3', '0', '4', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4'], ['2', '0', '0', '2', '4']]

iteration: 4

Gen. 0 (0.00%): Max/Min/Avg Fitness(Raw) [5.31(9.00)/3.57(0.00)/4.43(4.42)]

Gen. 10 (50.00%): Max/Min/Avg Fitness(Raw) [8.79(10.00)/4.96(3.00)/7.33(7.33)]

Gen. 20 (100.00%): Max/Min/Avg Fitness(Raw) [8.74(10.00)/4.98(3.00)/7.29(7.29)]

Total time elapsed: 480.457 seconds.

[['1', '1', '4', '1', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '1', '0', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '3', '0', '4'], ['2', '0', '3', '0', '4']]

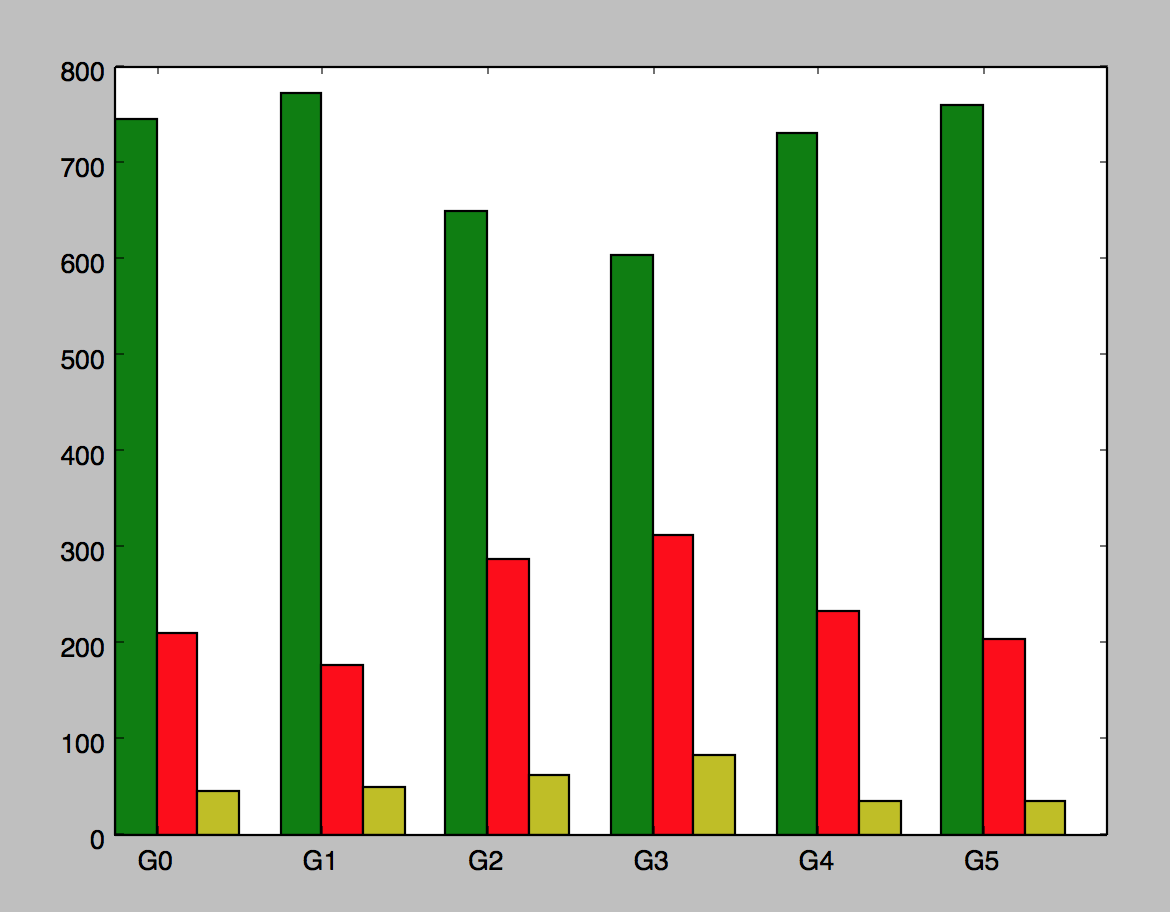
Best list: [['2', '0', '0', '2', '4'], ['3', '2', '3', '2', '4'], ['0', '0', '0', '3', '4'], ['2', '1', '3', '3', '0'], ['2', '0', '0', '2', '4'], ['2', '0', '3', '0', '4']]

Compare to best list of swap mutation:

[['1', '2', '3', '2', '4'],['0', '0', '0', '2', '4'],['0', '3', '2', '2', '4'],['3', '0', '2', '0', '2'],['0', '0', '2', '0', '4'],['2', '3', '3', '2', '0']]

several 2,4 in the last positions: advanced build and random

integer substitute runs faster than swap



Working on testing the entire strategy space to see if I can find an absolute minimum for n=15 on this one pickled graph.

Looking for top 50 for n=15 across the entire strategy space

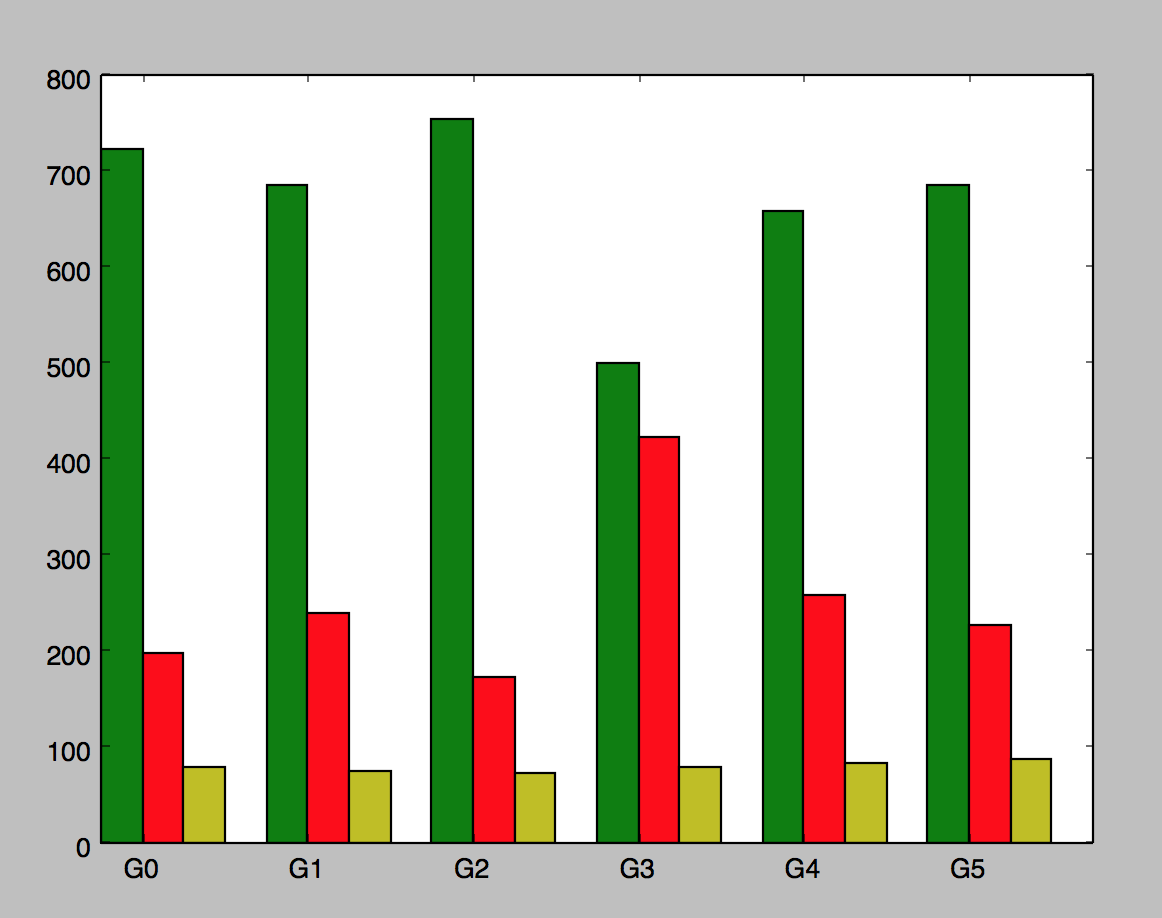
[[809, 159, 32, ('adv-build', 'build', 'adv-build', 'adv-build', 'random')], [790, 180, 30, ('adv-build', 'adv-build', 'adv-build', 'adv-build', 'random')], [783, 185, 32, ('adv-build', 'block', 'adv-build', 'adv-build', 'random')], [782, 182, 36, ('adv-build', 'adv-block', 'build', 'adv-build', 'random')], [781, 192, 27, ('adv-block', 'build', 'adv-build', 'adv-build', 'random')], [776, 191, 33, ('adv-block', 'adv-build', 'adv-build', 'adv-build', 'random')], [775, 200, 25, ('adv-build', 'adv-block', 'adv-build', 'adv-build', 'random')], [773, 198, 29, ('adv-block', 'adv-build', 'build', 'adv-build', 'random')], [769, 199, 32, ('adv-build', 'build', 'adv-block', 'adv-build', 'random')], [764, 202, 34, ('build', 'adv-build', 'adv-build', 'adv-build', 'random')], [764, 202, 34, ('block', 'adv-build', 'adv-build', 'adv-build', 'random')], [764, 201, 35, ('block', 'build', 'adv-build', 'adv-build', 'random')], [761, 205, 34, ('adv-build', 'adv-build', 'block', 'adv-build', 'random')], [758, 214, 28, ('adv-build', 'adv-build', 'adv-build', 'build', 'random')], [757, 214, 29, ('build', 'adv-block', 'adv-build', 'adv-build', 'random')], [757, 214, 29, ('adv-block', 'build', 'build', 'adv-build', 'random')], [756, 202, 42, ('adv-build', 'adv-build', 'adv-block', 'adv-build', 'random')], [754, 211, 35, ('build', 'block', 'adv-build', 'adv-build', 'random')], [751, 217, 32, ('build', 'build', 'adv-build', 'adv-build', 'random')], [750, 219, 31, ('block', 'adv-block', 'adv-build', 'adv-build', 'random')], [750, 212, 38, ('block', 'adv-build', 'build', 'adv-build', 'random')], [749, 224, 27, ('build', 'adv-build', 'build', 'adv-build', 'random')], [749, 217, 34, ('build', 'build', 'adv-block', 'adv-build', 'random')], [749, 213, 38, ('adv-build', 'adv-build', 'build', 'adv-build', 'random')], [749, 212, 39, ('build', 'adv-build', 'adv-block', 'adv-build', 'random')], [747, 225, 28, ('build', 'adv-block', 'build', 'adv-build', 'random')], [747, 216, 37, ('adv-block', 'build', 'adv-block', 'adv-build', 'random')], [747, 214, 39, ('block', 'adv-build', 'adv-build', 'build', 'random')], [745, 228, 27, ('adv-block', 'adv-block', 'adv-build', 'adv-build', 'random')], [745, 218, 37, ('adv-build', 'adv-block', 'adv-build', 'random', 'adv-build')], [744, 222, 34, ('adv-build', 'adv-block', 'adv-build', 'build', 'random')], [744, 218, 38, ('adv-build', 'adv-build', 'adv-block', 'build', 'random')], [738, 228, 34, ('adv-block', 'adv-build', 'adv-build', 'build', 'random')], [737, 248, 15, ('adv-block', 'adv-build', 'build', 'build', 'random')], [737, 236, 27, ('adv-build', 'build', 'build', 'adv-build', 'random')], [737, 224, 39, ('adv-build', 'block', 'adv-build', 'build', 'random')], [737, 222, 41, ('adv-build', 'adv-block', 'adv-block', 'adv-build', 'random')], [736, 230, 34, ('adv-block', 'adv-build', 'adv-build', 'random', 'adv-build')], [735, 238, 27, ('block', 'adv-block', 'build', 'adv-build', 'random')], [735, 234, 31, ('block', 'build', 'build', 'adv-build', 'random')], [734, 231, 35, ('build', 'adv-build', 'adv-block', 'build', 'random')], [733, 240, 27, ('adv-build', 'build', 'adv-build', 'build', 'random')], [733, 228, 39, ('block', 'adv-build', 'adv-block', 'adv-build', 'random')], [728, 232, 40, ('adv-block', 'adv-build', 'adv-block', 'adv-build', 'random')], [727, 241, 32, ('block', 'build', 'adv-build', 'build', 'random')], [726, 243, 31, ('adv-block', 'adv-block', 'adv-build', 'build', 'random')], [726, 237, 37, ('adv-block', 'block', 'adv-build', 'adv-build', 'random')], [724, 250, 26, ('build', 'build', 'build', 'adv-build', 'random')], [723, 242, 35, ('block', 'adv-build', 'adv-build', 'random', 'adv-build')], [723, 237, 40, ('adv-build', 'adv-build', 'adv-build', 'adv-block', 'random')]]

3.11.18

I want to repeat the experiments done above for the 6 evolutions involving

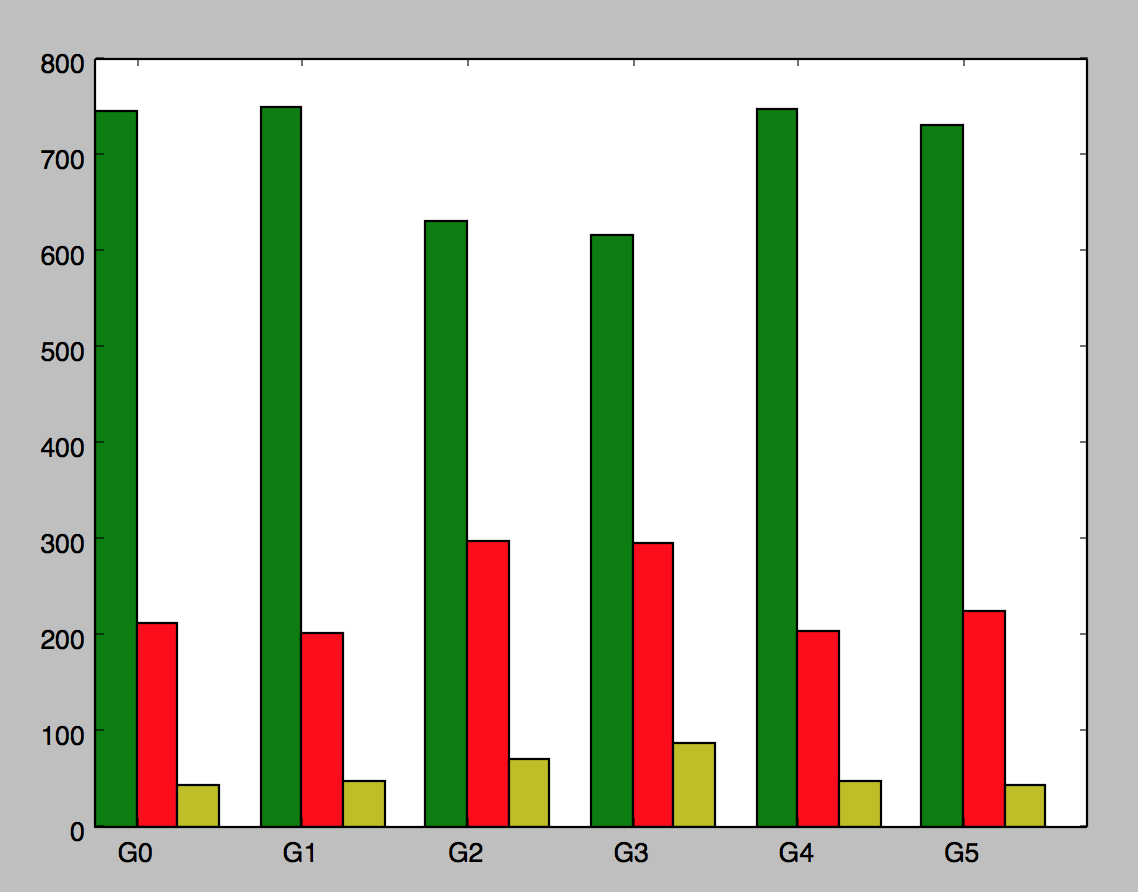
Swap mutation:

[['1', '2', '3', '2', '4'],['0', '0', '0', '2', '4'],['0', '3', '2', '2', '4'],['3', '0', '2', '0', '2'],['0', '0', '2', '0', '4'],['2', '3', '3', '2', '0']]



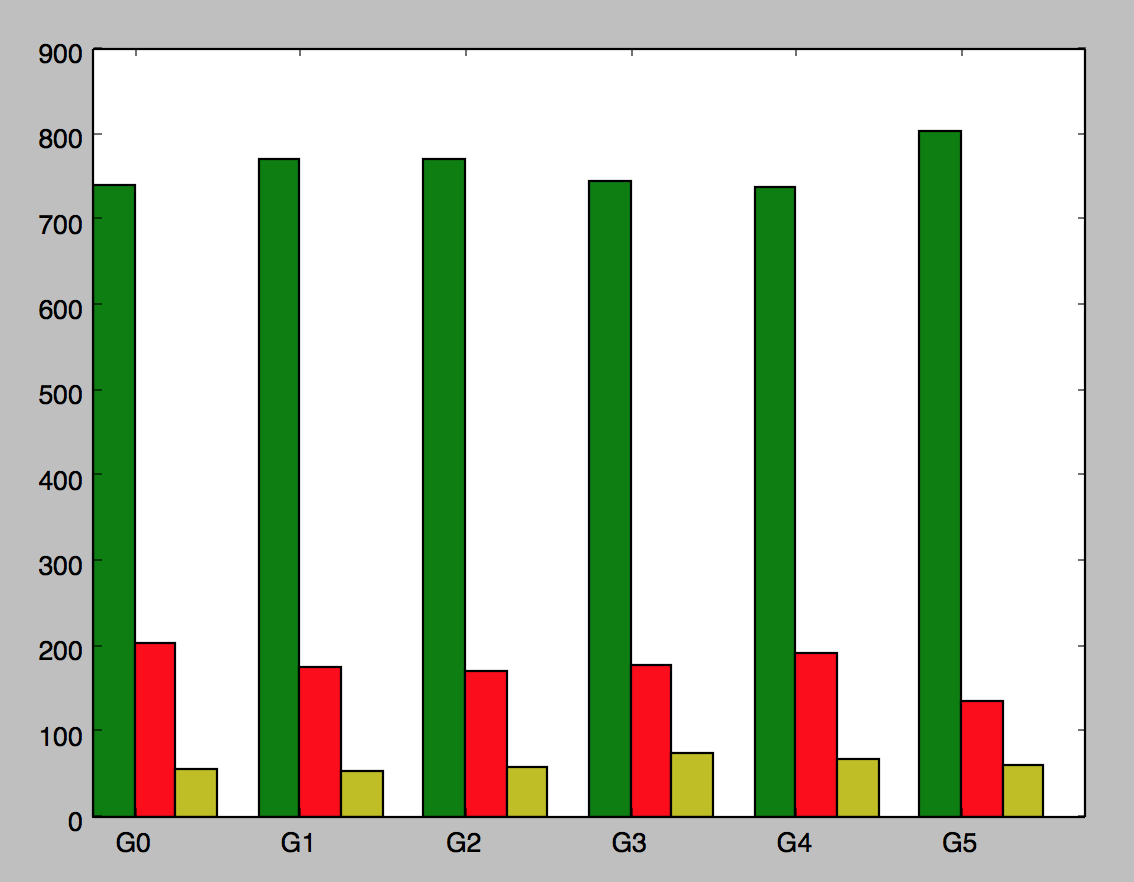
integer range mutation

[['2', '0', '0', '2', '4'], ['3', '2', '3', '2', '4'], ['0', '0', '0', '3', '4'], ['2', '1', '3', '3', '0'], ['2', '0', '0', '2', '4'], ['2', '0', '3', '0', '4']]

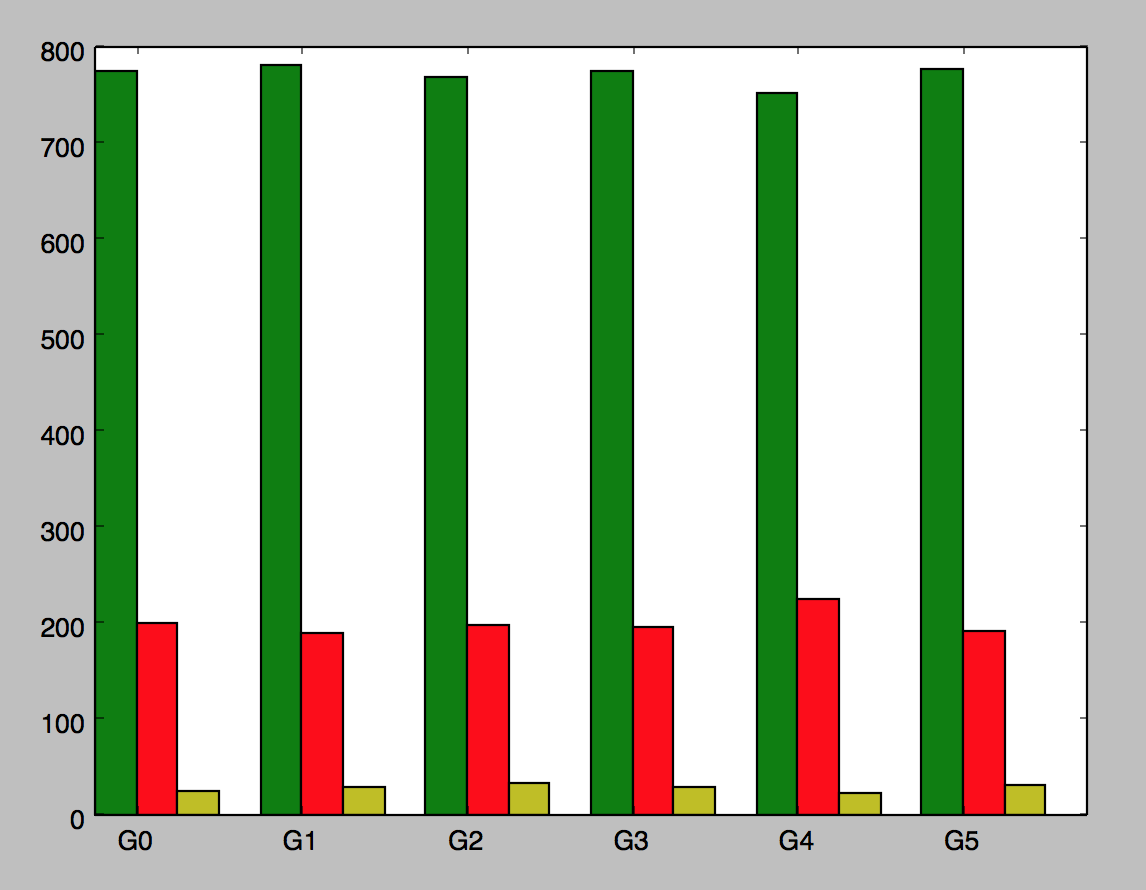


And compare to alleged top 6 across the space

[[2, 0, 2, 2, 4], [2, 2, 2, 2, 4], [2, 1, 2, 2, 4], [2, 3, 0, 2, 4], [3, 0, 2, 2, 4], [3, 2, 2, 2, 4]]



Top 6 of 50 played a 1000 games on a different pickled graph (1 instead of 0)’



Question if we double the intervals does integer range mutation still beat swap mutation?

Twenty generations

Ran 6 evolutions for swap on fixed graph:

[['0', '2', '1', '0', '1', '3', '4', '1', '0', '2'], ['1', '0', '1', '1', '3', '3', '0', '2', '0', '4'], ['1', '0', '0', '4', '1', '1', '0', '2', '2', '1'], ['1', '0', '2', '1', '0', '2', '2', '0', '0', '3'], ['3', '0', '2', '1', '1', '0', '0', '4', '2', '1'], ['2', '3', '0', '3', '2', '0', '4', '4', '1', '0']]

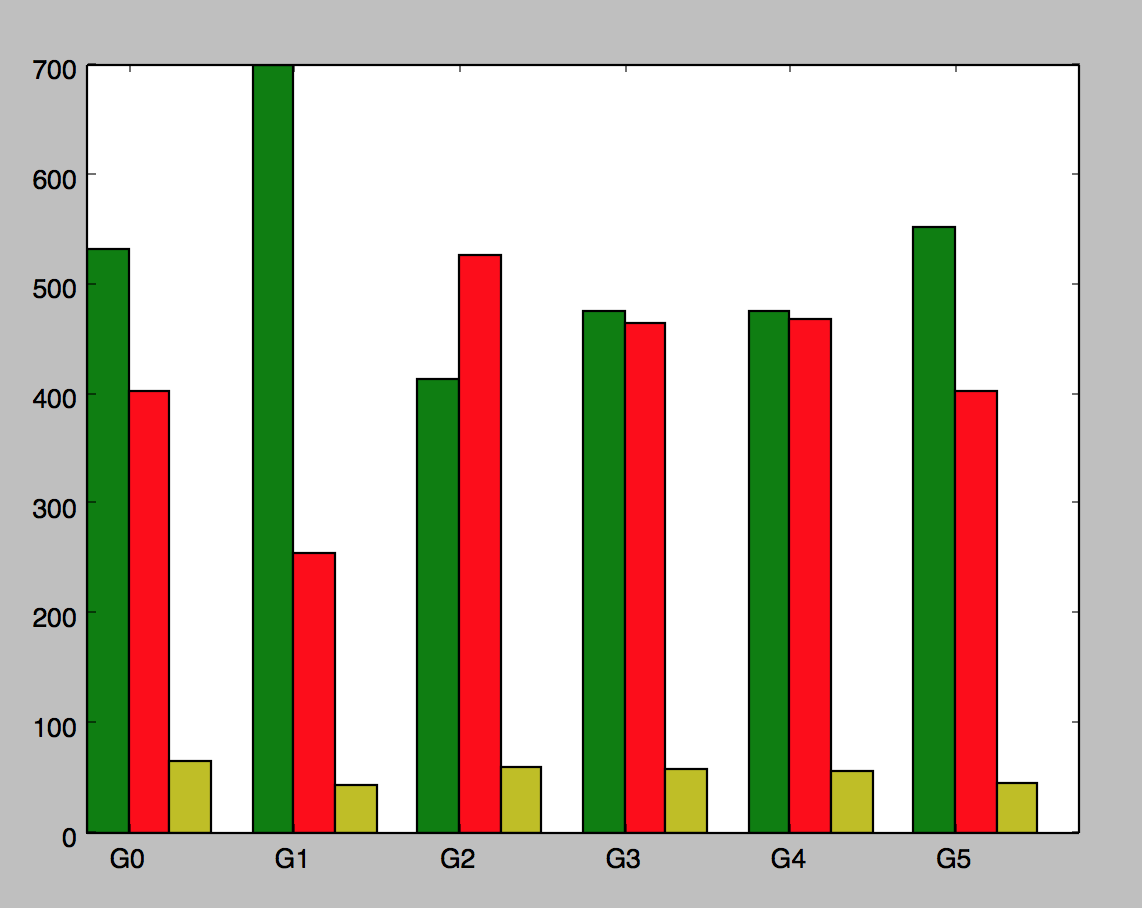
Ran 6 evolution for integer range on same fixed graph:

[['2', '0', '1', '4', '0', '0', '3', '3', '4', '0'], ['2', '2', '0', '3', '2', '0', '2', '0', '2', '4'], ['3', '0', '0', '2', '2', '3', '0', '0', '2', '4'], ['0', '3', '2', '0', '3', '1', '4', '0', '4', '0'], ['1', '3', '2', '1', '2', '2', '3', '4', '2', '4'], ['0', '0', '4', '1', '0', '3', '2', '4', '4', '0']]

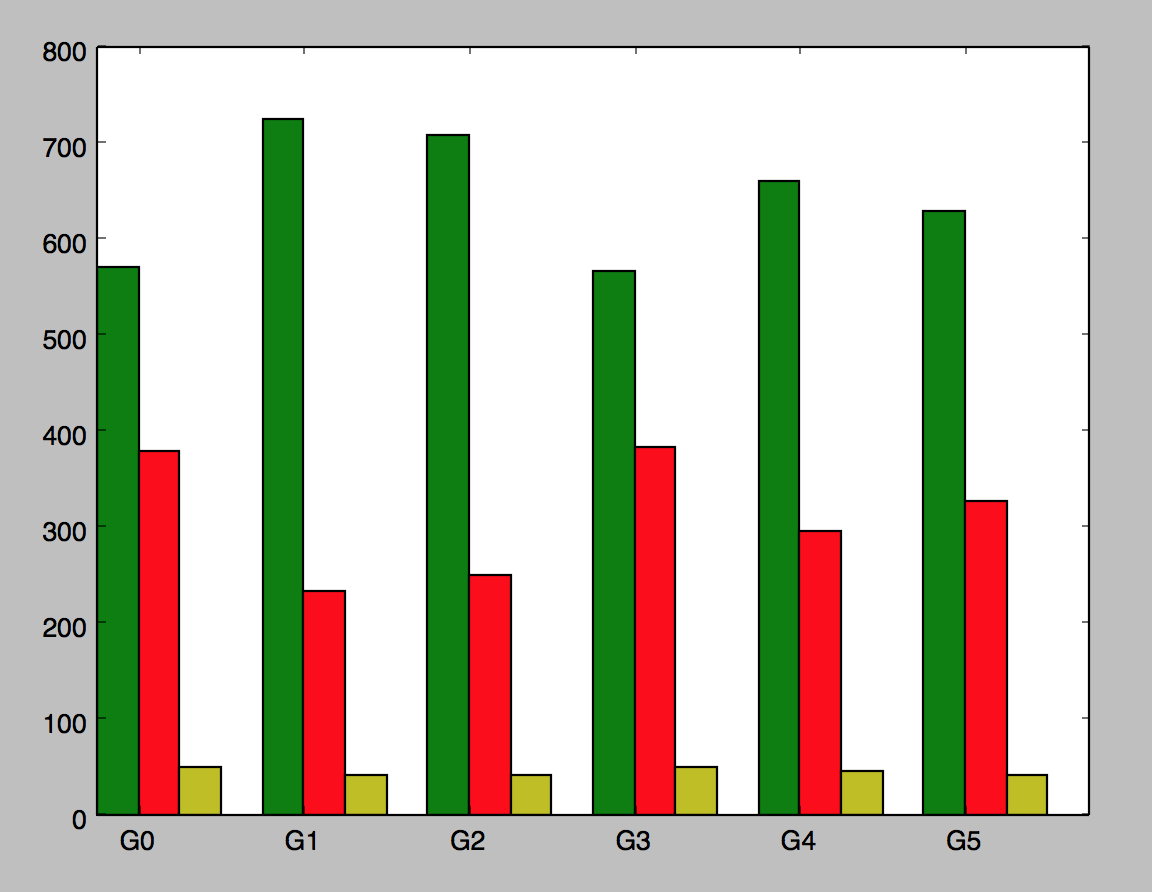
n=1000

now run comparison convergence

Swap



Integer range:



IR still much better than swap

NOW Double number of generations and population to see if this helps performance

Number individuals in population 160

Number of Generations: 40

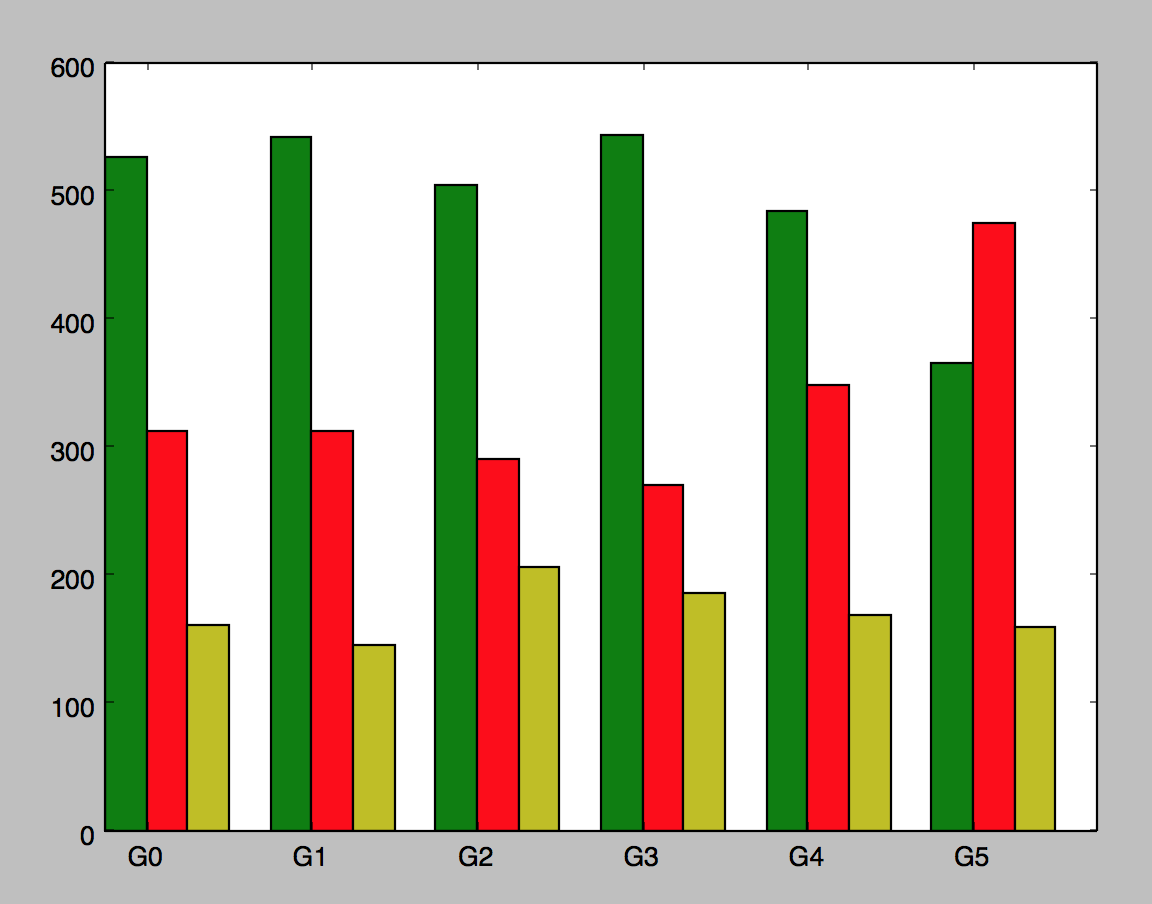
Swap Results:

[['2', '2', '1', '2', '2', '2', '3', '0', '2', '4'], ['2', '0', '2', '3', '0', '2', '1', '3', '4', '2'], ['1', '0', '0', '2', '1', '2', '4', '2', '2', '4'], ['2', '3', '2', '3', '2', '2', '3', '2', '4', '4'], ['3', '2', '1', '0', '0', '0', '1', '0', '2', '4'], ['0', '4', '0', '1', '4', '0', '3', '3', '2', '4']]

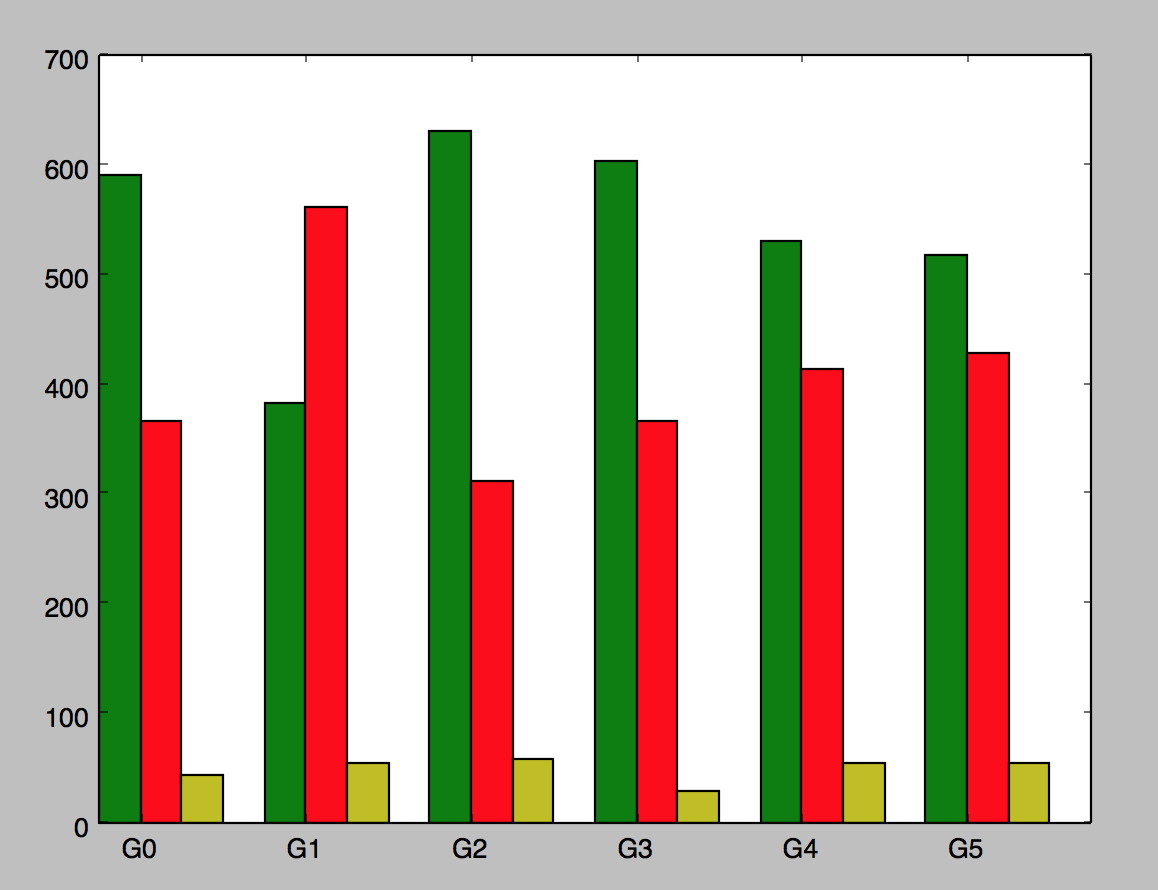
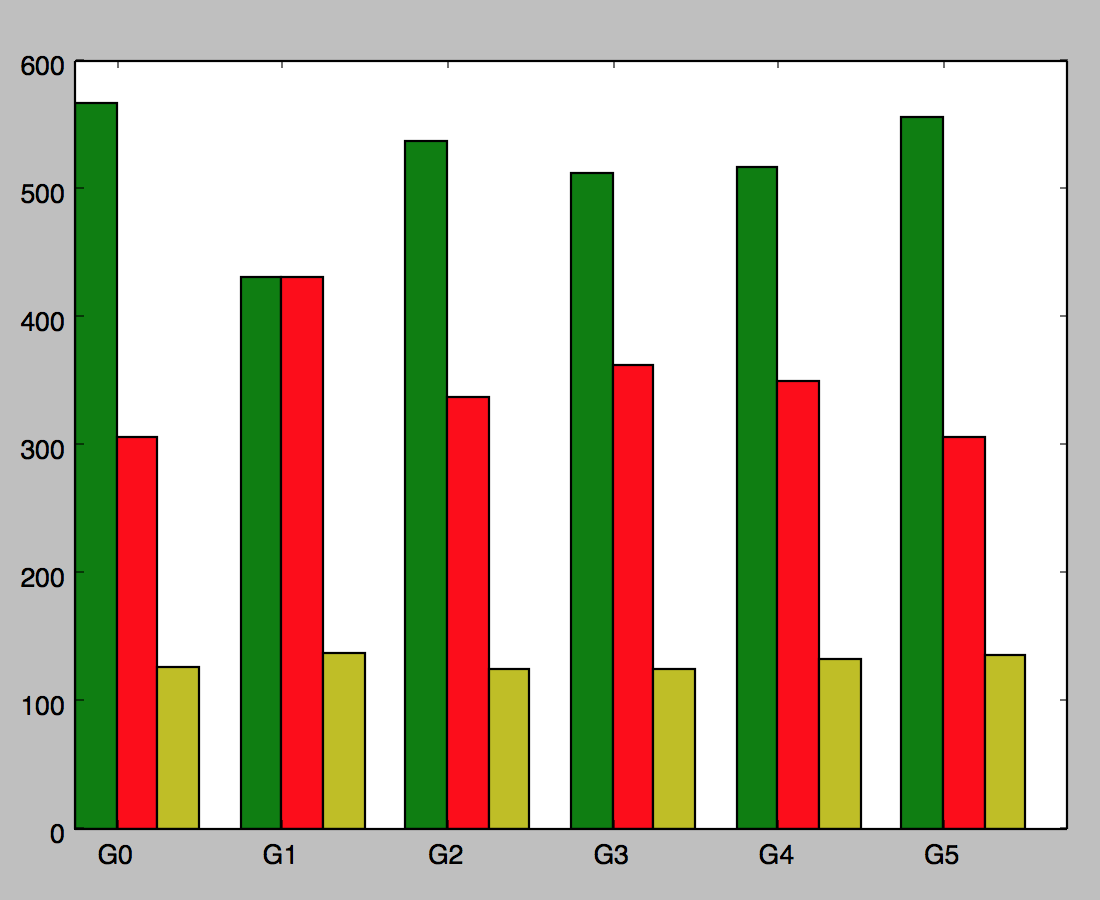
IR Results:

[['2', '0', '3', '3', '0', '3', '4', '0', '0', '0'], ['1', '0', '0', '1', '1', '1', '3', '1', '4', '1'], ['2', '3', '2', '2', '2', '3', '3', '0', '3', '0'], ['0', '0', '0', '2', '1', '2', '4', '0', '2', '0'], ['2', '1', '0', '2', '1', '3', '2', '1', '1', '2'], ['0', '3', '3', '3', '2', '3', '3', '0', '1', '3']]

Swap graph



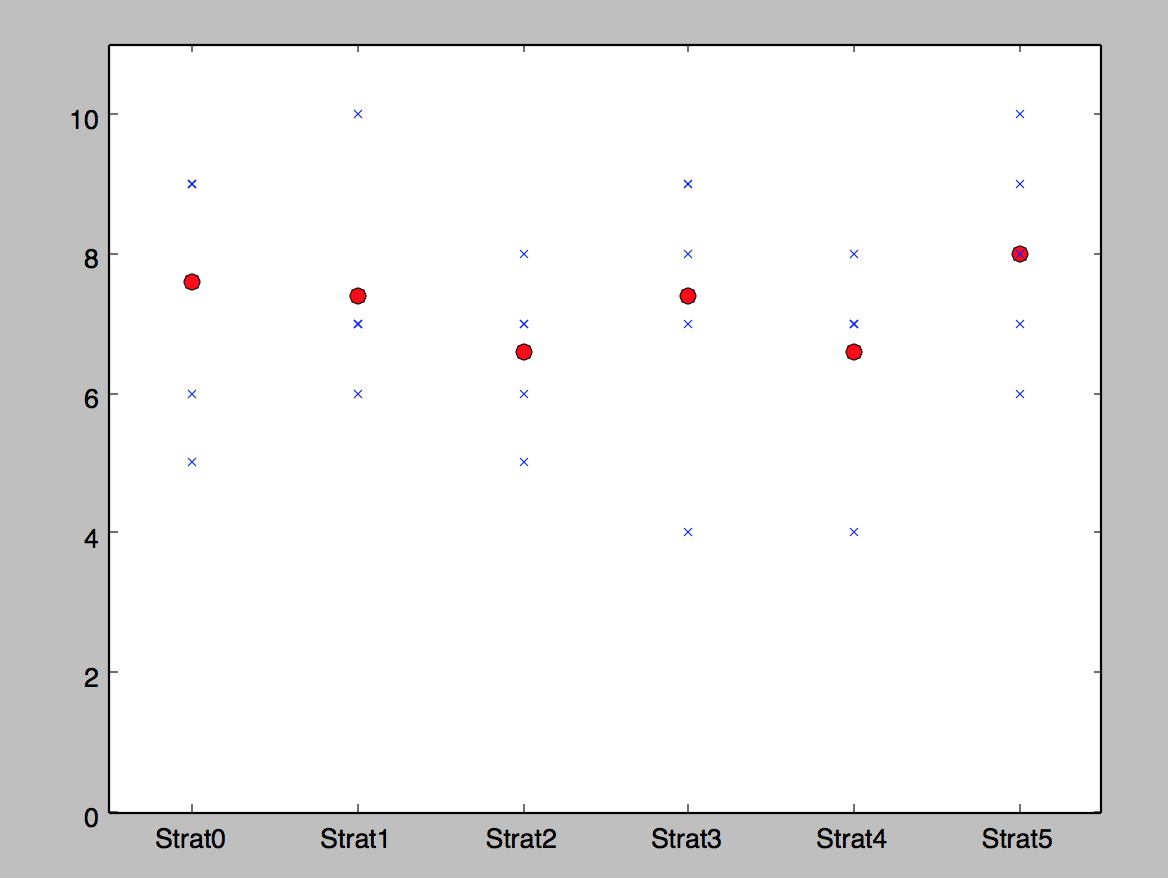
IR graph



new topic: taking top six strategies for graph 0 taking s samples of them running n times

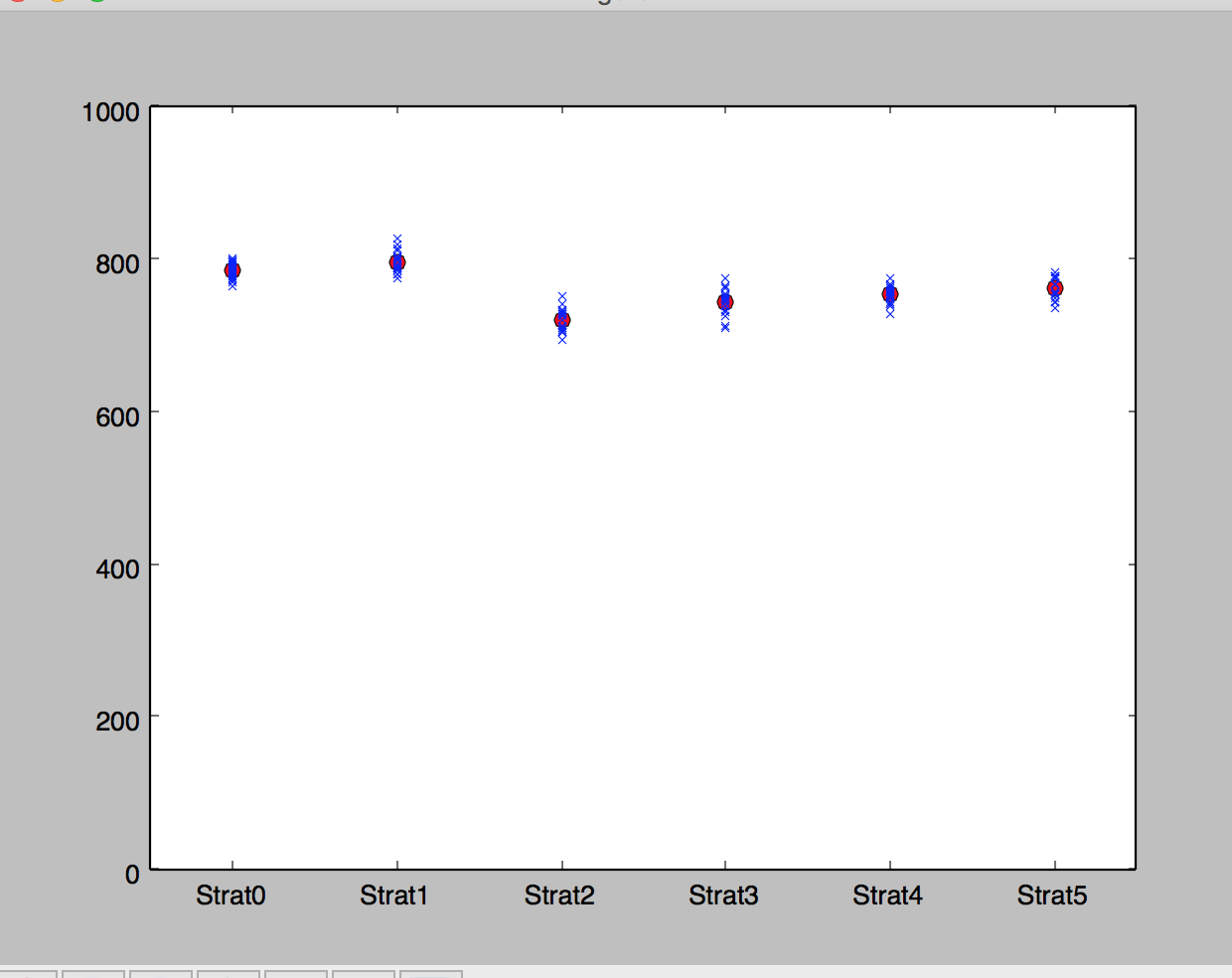
n = 10

s = 5



S = 20

N = 1000



With this in mind, I wanted to improve the evolution on fixed graphs, so I decided to increase the loop in the fitness function.

Gen. 0 (0.00%): Max/Min/Avg Fitness(Raw) [28.15(38.00)/19.12(10.00)/23.46(23.46)]

Gen. 10 (66.67%): Max/Min/Avg Fitness(Raw) [32.50(42.00)/23.06(16.00)/27.09(27.09)]

Gen. 15 (100.00%): Max/Min/Avg Fitness(Raw) [31.28(43.00)/23.27(17.00)/26.06(26.06)]

Total time elapsed: 545.283 seconds.

[['2', '4', '0', '1', '3'], ['0', '3', '4', '4', '0'], ['0', '3', '4', '4', '0'], ['0', '3', '4', '4', '0'], ['0', '3', '4', '4', '0'], ['0', '3', '4', '4', '0'], ['0', '3', '4', '4', '0'], ['0', '3', '4', '4', '0'], ['0', '3', '4', '4', '0'], ['0', '3', '4', '4', '0'], ['0', '3', '4', '4', '0'], ['0', '3', '4', '4', '0'], ['0', '3', '4', '4', '0'], ['0', '3', '4', '4', '0'], ['0', '3', '4', '4', '0'], ['0', '3', '4', '4', '0']]

this is with 100 graphs. Not super great.

3/23/18

Task: increase number of iterations in fitness function to 1000 in fixed graphs

N=15 graph is index 0 pickled1000n15

Results :

Gen. 0 (0.00%): Max/Min/Avg Fitness(Raw) [553.50(790.00)/384.57(188.00)/461.25(461.25)]

Gen. 20 (100.00%): Max/Min/Avg Fitness(Raw) [938.48(828.00)/489.03(696.00)/782.06(782.06)]

Total time elapsed: 44761.123 seconds.

[['1', '1', '1', '4', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4'], ['3', '2', '2', '2', '4']]

NOTES:

>took so long! Is it deepy copy? Run it without deep copy

> this is #6 in the top six strategies

> basically optimal

Ran again without deep copy to compare times:

Results:

* chromosome
* time comparisons