ECM3412 – Nature-Inspired Computation

Assignment 1: Bin-Packing with an Evolutionary Algorithm

Method

In this assignment, I have implemented six versions of an evolutionary algorithm (EA) and applied them to two bin-packing problems: BPP1 (10 bins) and BPP2 (100 bins). In each problem, 500 items (weighing a different amount) must be evenly distributed amongst the bins. In all versions of the EA, a random initial population is generated. Binary tournament selection is used (twice) to select two parents. The fitness of a given solution is evaluated by calculating the difference between the lightest and heaviest bins; better solutions have a smaller difference. The parents are altered in some way according to the EA parameters, and weakest replacement is used to determine whether new solutions should overwrite the worst existing solutions in the population. This continues until 10,000 fitness evaluations have been reached. The parameters altered between versions of the EA are: whether or not single-point crossover is used, the type of multi-gene mutation operator used (if any), and the population size (10 or 100 solutions). Each version of the algorithm was run five times, using different random number seeds for each trial. To allow for an objective comparison between the algorithms, the same (differing) seeds were used in each experiment. At the end of each trial, the fitness of the best solution in the population was recorded, and the average (mean) best fitness of the five trials was calculated.

During the trials, additional data was collected for further experiments. In each of the five trials, the fitness of the best solution was recorded at intervals of 1000 fitness evaluations (i.e. [1, 1000, … , 9000, 10000]). The mean of each interval was calculated and used to plot a line graph comparing how the average best recorded fitness changes over time in each of the algorithms, as opposed to looking solely at the final averages. In each trial, the best and worst fitness of the initial population is recorded. These values are then used to plot a grouped bar chart comparing the average difference between the initial and final best/worst solutions. The graph intends to contextualise changes in population fitness and the differences in search space. Lastly, the fitness of solutions in each of the final populations from the five trials are gathered and displayed in a violin plot. This provides insight into how the fitness of final solutions are distributed for each of the algorithms.

Results

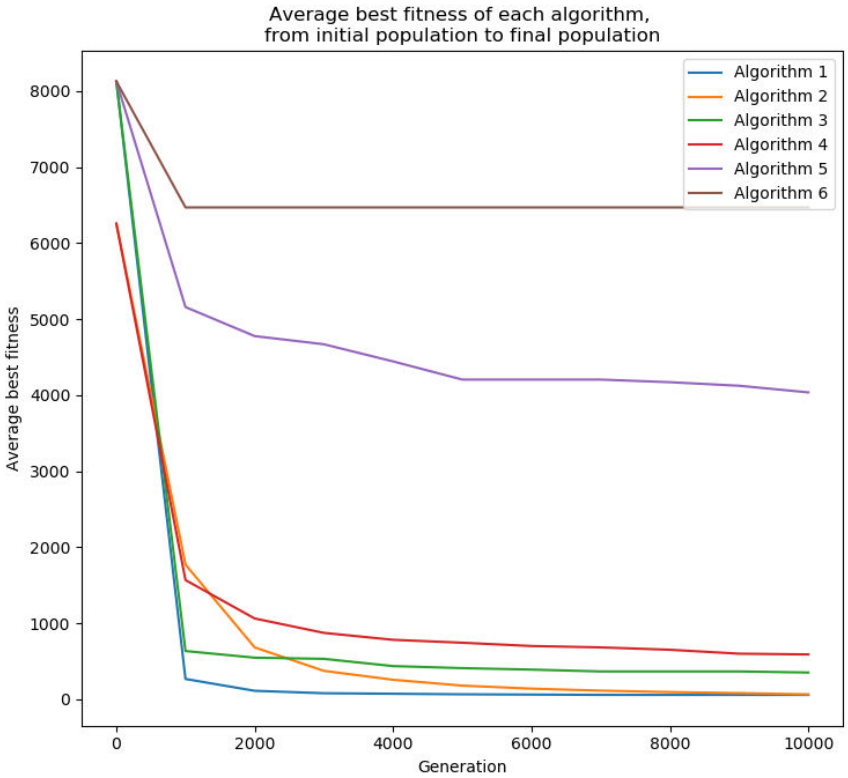
BPP1 Results

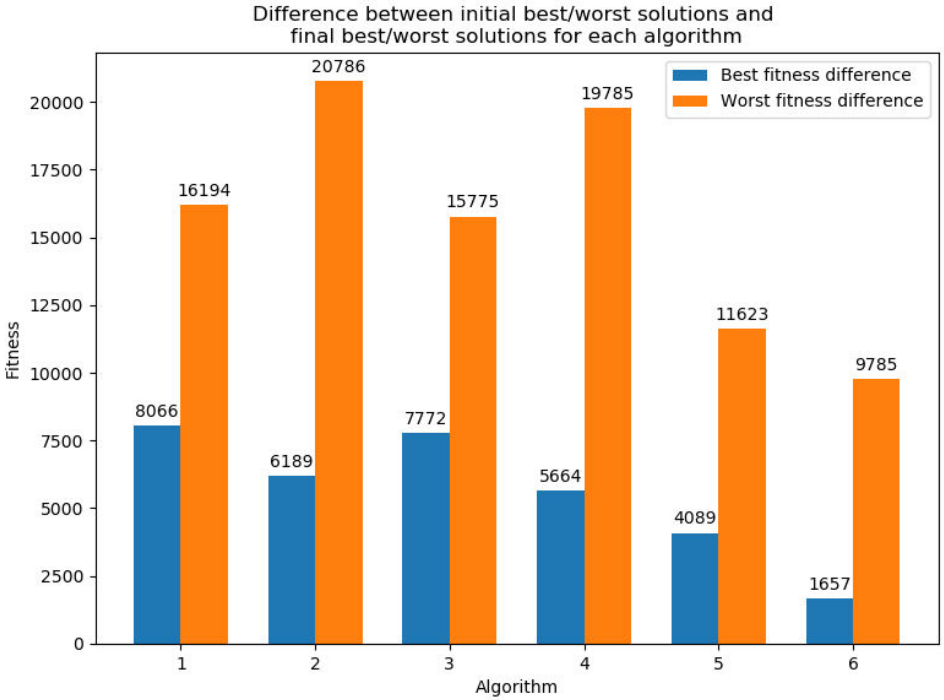
Algorithm	Seed 1	Seed 2	Seed 3	Seed 4	Seed 5	Average
1	104	78	26	40	46	58.8
2	82	56	64	64	64	66.0
3	404	370	356	294	342	353.2
4	522	662	626	608	536	590.8
5	3242	3394	4566	4586	4394	4036.4
6	6220	8542	7082	4114	6384	6468.4

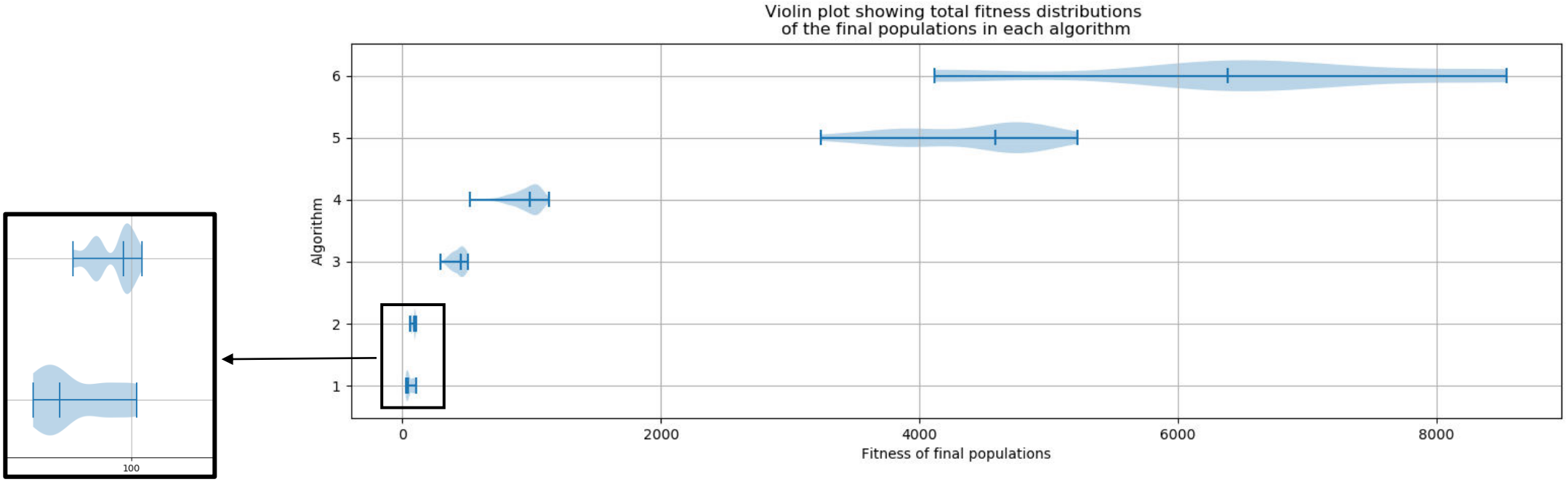
BPP2 Results

Algorithm	Seed 1	Seed 2	Seed 3	Seed 4	Seed 5	Average
1	600436	493000	514848	504078	527018	527876.0
2	991246	762568	874996	759130	777432	833074.4
3	469212	506662	474868	473358	513702	487560.4
4	779510	892480	811986	793870	847314	825032.0
5	1559970	1676876	1692812	1626330	1678306	1646858.8
6	2073368	1840492	2076034	1825696	1755150	1914148.0

BPP1 Graphs

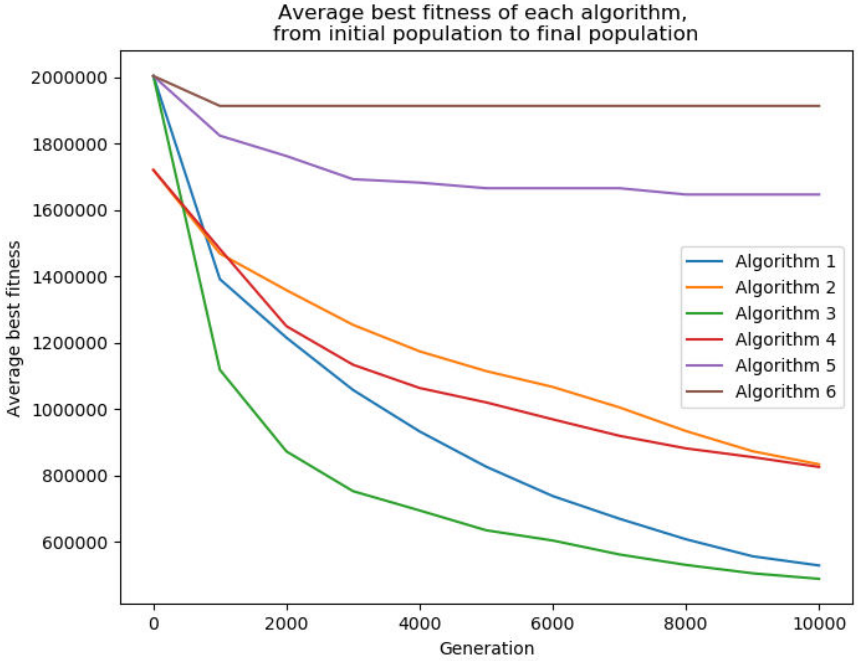

Average best fitness of each algorithm,
from initial population to final population
Algorithm 1
Algorithm 2
Algorithm 3
Algorithm 4
Algorithm 5
Algorithm 6
Average best fitness
Generation
0
1000
2000
3000
4000
5000
6000
7000
8000
0
2000
4000
6000
8000
10000

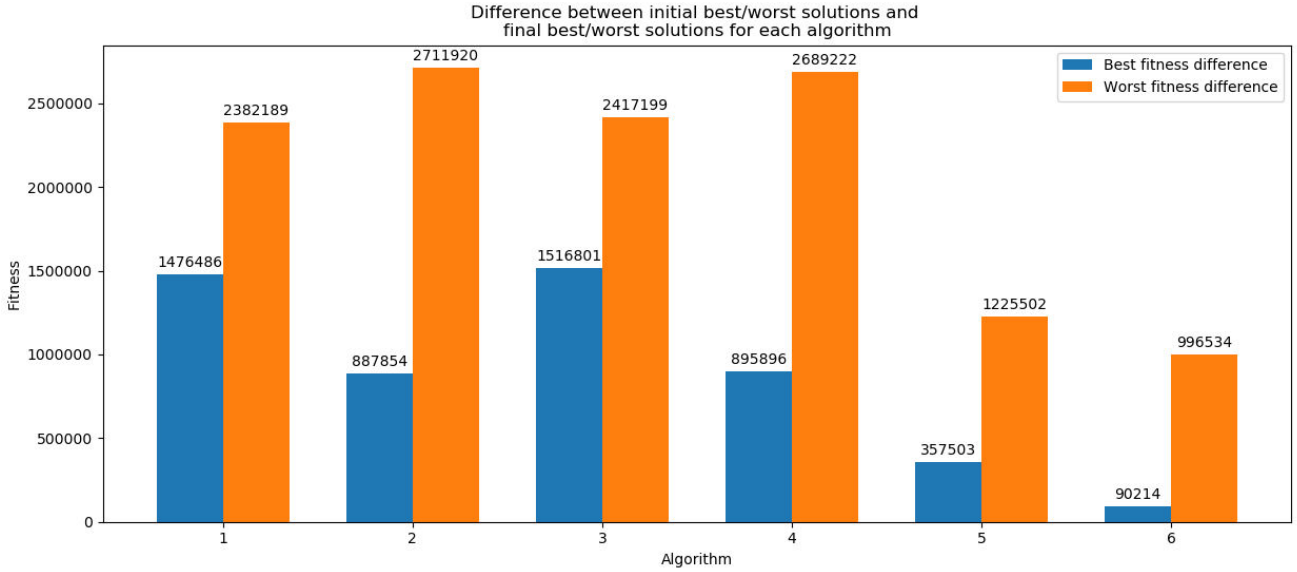

Difference between initial best/worst solutions and
final best/worst solutions for each algorithm
Best fitness difference
Worst fitness difference
Fitness
Algorithm
8066
16194
6189
20786
7772
15775
5664
19785
4089
11623
1657
9785

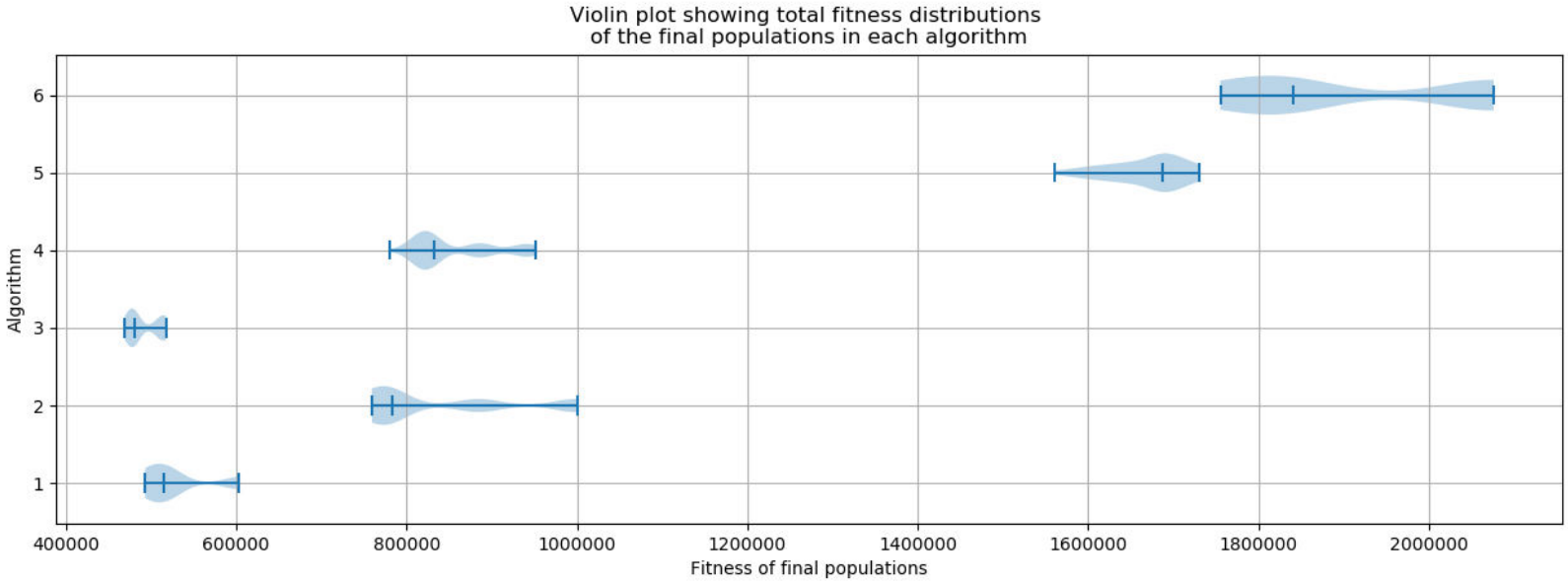

Violin plot showing total fitness distributions
of the final populations in each algorithm
Algorithm
Fitness of final populations
100

BPP2

Graphs


Average best fitness of each algorithm,
from initial population to final population
Average best fitness
Generation
Algorithm 1
Algorithm 2
Algorithm 3
Algorithm 4
Algorithm 5
Algorithm 6


Difference between initial best/worst solutions and
final best/worst solutions for each algorithm
Fitness
Algorithm
Best fitness difference
Worst fitness difference
1476486
2382189
887854
2711920
1516801
2417199
895896
2689222
357503
1225502
90214
996534


Violin plot showing total fitness distributions
of the final populations in each algorithm
Algorithm
Fitness of final populations

Question 1: Which combination of parameters produces the best results for BPP1 and BPP2?

BPP1: When considering the single best solution in the final population, algorithms with crossover and operator M1 produced the fittest solutions on average. Of these, Algorithm 1 (population size 10) produced fitter solutions on average than Algorithm 2 (population size 100). Looking at the line chart, we can see that Algorithm 1 obtains its fittest solution after around 2000 fitness evaluations, whilst Algorithm 2 requires around 8000 fitness evaluations. However, looking at the fitness difference between the initial and final best/worst solutions, and at the corresponding fitness distributions, we observe Algorithm 2 produced around the same worst solution fitness as Algorithm 1, despite having a larger search space.

BPP2: Considering the single best solution in the final population, Algorithm 3 (crossover, M5, population size 10) produced the fittest solutions on average. Considering the entire final population, the violin plot shows Algorithm 3 only slightly outperformed Algorithm 1 (crossover, M1, population size 10). This is confirmed in the other graphs; looking at the line chart we see Algorithm 1 and 3 had the same initial average best fitness, and on the bar chart we see the difference between their initial best and worst and final best and worst solutions are very similar.

Question 2: Why do you think this is the case?

In BPP1 there are only 10 bins, so randomly generated solutions in the initial population are more likely to have a better fitness than if there were 100 bins. So, in BPP1 the mutation operator M5 introduces too much variation and results in new solutions that are not fitter than the least fit in the initial population. M1 introduces only a slight variation, so the initial population can be better optimised when there are 10 bins. In BPP2, the search space is large enough for operator M5 to be more effective than M1. M5 traverses more of the search space and therefore finds better solutions. Combinations of crossover and mutation are better than either on their own; crossover between parents increases the likelihood of a child solution that is better than its parents, and mutation ensures new areas of the search space are explored.

Question 3: What was the effect when you removed mutation? What about crossover?

In both BPP1 and BPP2, removing the mutation and performing only crossover (i.e. Algorithm 6) produced the least fit best solution. In the line charts, we see it finds its best solution after around 1000 fitness evaluations. Looking at the violin plot we can see it also produced the most spread population (in terms of fitness). This is justified in the bar chart, where we observe it has the least difference between the initial best and worst solutions and final best and worst solutions. Removing the crossover and performing only mutation operator M5 (i.e. Algorithm 5) produced the second least fit best solution, in both BPP1 and BPP2. In BPP1, both Algorithm 5 and 6 have significantly more spread fitness distributions than the other Algorithms (some combination of mutation and crossover). Looking at both line charts, we can clearly see that a combination of crossover and mutation produces significantly more fit solutions than crossover or mutation alone. However, mutation alone produces significantly more fit solutions than crossover alone.

Question 4: Which other nature-inspired algorithms might be effective in optimising the BPP problems?

An Ant Colony Optimisation algorithm could be used, where each ant represents one of the 500 items and pheromones influence whether or not an ant enters a bin. As a bin becomes more populated, the pheromone strength associated with the bin weakens.

A Genetic Programming algorithm might also be effective in optimising the BPP problems. It could evolve an entire program to solve the BPP problems, rather than evolving individual solutions to the problems as we have done with the EA in this assignment.