Team 2 CS4341 Homework 2

William Edor, Tri Khuu, Peerapat Luxsuwong, Saraj Pirasmepulkul

1. Your approach to computing a fitness function, selection, child generation, crossover, and mutation. Also, what population size did you use? You may use the same approach to these for all 3 puzzles, but are not required to. Provide an example of evaluating a small population, selection, and generating a child for one set of parents. If your approach varies across problems, show an example of each.

Ans:

Population size: 500, by default both elitism and culling are used.

EVALUATION

Different callback functions are used to evaluate the fitness of populations for the three puzzles. It is also in the evaluation function that culling is executed. First of all, each member of the population is run through its callback function to assign a score to it. Then, a fitness value is calculated for each member of the population, based on its score. In puzzle 1, the score will always be zero or greater than zero. In puzzles 2 and 3, the score may be negative, so the inverse of the absolute value of the score is used in these cases. The resulting fitness value is then squared, to increase the chances of members with higher scores being selected in the selection process. a tiny amount, 0.00001 is added to differentiate bad strings from culled strings. Then, to prepare the population for the selection process, where roulette-wheel selection will be used, the members of the population were rearranged in order of fitness values.

It was after this that the lowest ~20% of the population were assigned 0 for their fitness values, to simulate culling. Then, the fitness values of each member of the population were reevaluated, so that they would also include the sum of each fitness value below it. After this, the fitness values of the population were finally normalized, so that the least fit member of the population would have a fitness value of ~0, and the most fit member of the population would have a fitness value of 1.

SELECTION

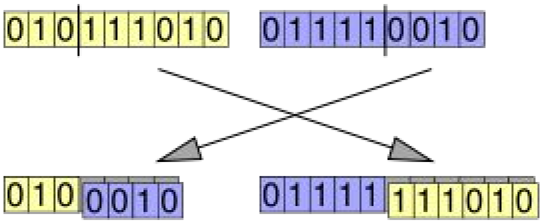
The same form of selection is done in all puzzles. A form of elitism is implemented in the selection process. It guarantees that the fittest individuals move on to the next stage which is crossover. A variable called FIT_NUM is used to determine the percentage of the fittest individuals that move to the crossover stage. FIT_NUM calculates this percentage but dividing the population size by 5 and adding 1, to approximate an elite population of ~20%. The rest of the population is selected to move to the next stage if their fitness value, provided from the evaluation stage, is higher than a cutoff value which is a random value between 0 and 1. After each selection of a string, a new cutoff value is determined to check if the next string is qualified to be selected.

For a population size of 500, the FIT_NUM value is 101. The first 101 strings are then placed into the array of selected strings. Strings from the remaining 299 are chosen if they have fitness values higher than the random cutoff value generated after each placement of a qualified string in the array of selected strings.

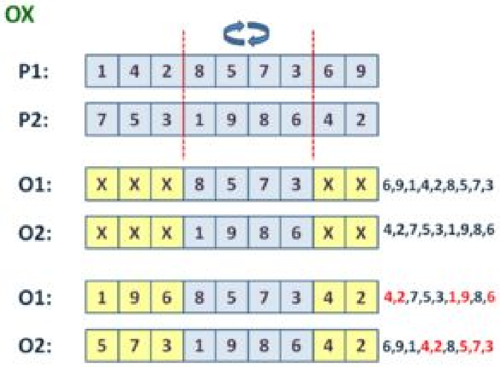
CROSSOVER

The way that the crossover is implemented is the same for puzzle 1 and 3, but is different than puzzle 2's method.

In puzzle 1 and 3, we have a temperature that decreases over time that determine the probability that crossover will happen for each element in the population. When two elements are chosen for crossover, two new elements are born by appending the first part of one parent to the tail of the other parent. The indexes at which the parents are cut are randomly generated.



In puzzle 3, we also have a temperature that decreases over time that determine the probability that crossover. However, partially crossover is implemented instead. When two parents are chosen for crossover, the mid-body of a parent is appended to the head and tail of the other one. To ensure that no gene is duplicated, only the gene that are not added to the element is picked.


OX
P1: 1 4 2 8 5 7 3 6 9
P2: 7 5 3 1 9 8 6 4 2
O1: X X X 8 5 7 3 X X 6,9,1,4,2,8,5,7,3
O2: X X X 1 9 8 6 X X 4,2,7,5,3,1,9,8,6
O1: 1 9 6 8 5 7 3 4 2 4,2,7,5,3,1,9,8,6
O2: 5 7 3 1 9 8 6 4 2 6,9,1,4,2,8,5,7,3

MUTATION

The way that the mutation is calculated is the same for puzzles 1 and 3, but is different than puzzle 2's method. In puzzles 1 and 3, the mutation was first calculated by creating a new random gene in the genes through new_gene. A new string_buf is then created. Next, a random number was generated whose bound is the size of the length of the string. This section of the string (gene) was then cut and added to the string_buf. The randomly created number is then added to the end of the gene list thus far, and then the remaining sections of the gene after the index is then added to the string_buf. In other words, a random index on the gene sequence was replaced by a randomly generated number as a new string called string_buf. This mutated string_buf is then added to the mutated population, mutated_pop.

For example, if the string of number (genes) is 1, 2, 3, 4, 5, if the randomly generated number is 9, and the randomly generated index is 2, then string_buf would first become 1, 2. Next, the randomly generated number 9 would be added, making string_buf 1,2,9. Lastly, 4 and 5 would be appended, forming string_buf = 1,2,9,4,5.

In puzzle 2, the mutate function works differently. In this case, the mutate function would swap out a number in Bin 1 and Bin 2 with a number in Bin 3 from the same index, which was randomly generated. The reason that this swapping is used is because Bin 3 is not used anyways, and the size of the bins are fixed to 10. First of all, two random ints called m_index1 and m_index2 are created that specifies the index of the number on Bin 1 and Bin 2 that would be swapped with a number in Bin 3. Bin 1 ranges from index 0 to 9, Bin 2 ranges from index 10 to 19, and Bin 3 ranges from 20 to 29. If m_index1 was 7 and m_index2 was 3, for example, the number at the 7th index in Bin 1 would be swapped with the number at the 7th index of Bin 3, and the number at the 3rd index in Bin 2 would be swapped with the number at the 3rd index in Bin 3. Consider this example, of a series of alphabet that represents the 30 numbers:

A,B,C,D,E,F,**G**,H,I,J   K,L,M,**N**,O,P,Q,R,S,T   U,V,W,**Y**,X,Z,**AA**,BB,CC,DD

After the mutation function, this series of alphabet would become:

A,B,C,D,E,F,**AA**,H,I,J   K,L,M,**Y**,O,P,Q,R,S,T   U,V,W,**N**,X,Z,**G**,BB,CC,DD

2. How you handled the problem of illegal children for crossovers. Provide an example, or explain how your representation prevented the problem from occurring.

Ans:

Illegal children were left alone, as they would be assigned scores of 0 in the evaluation phase if they did not follow the rules of the puzzle. In puzzle 2, however, we ensured that the size of each child was a size of 30 through a specific type of crossover.

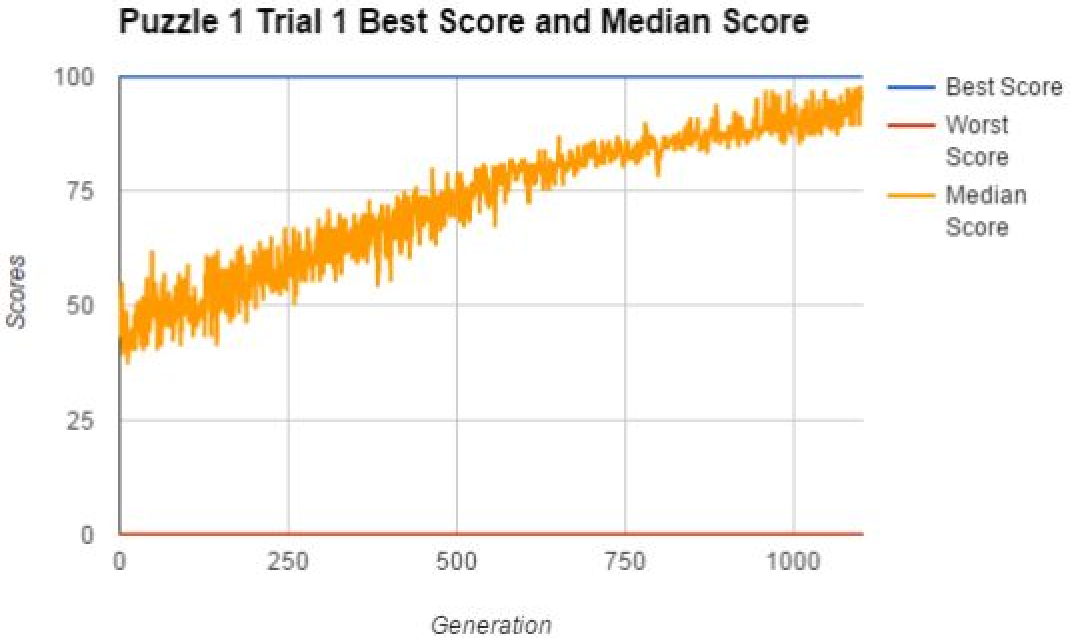
3. For each of the 3 puzzles, provide a graph showing how solution quality varies as the number of generations increases:

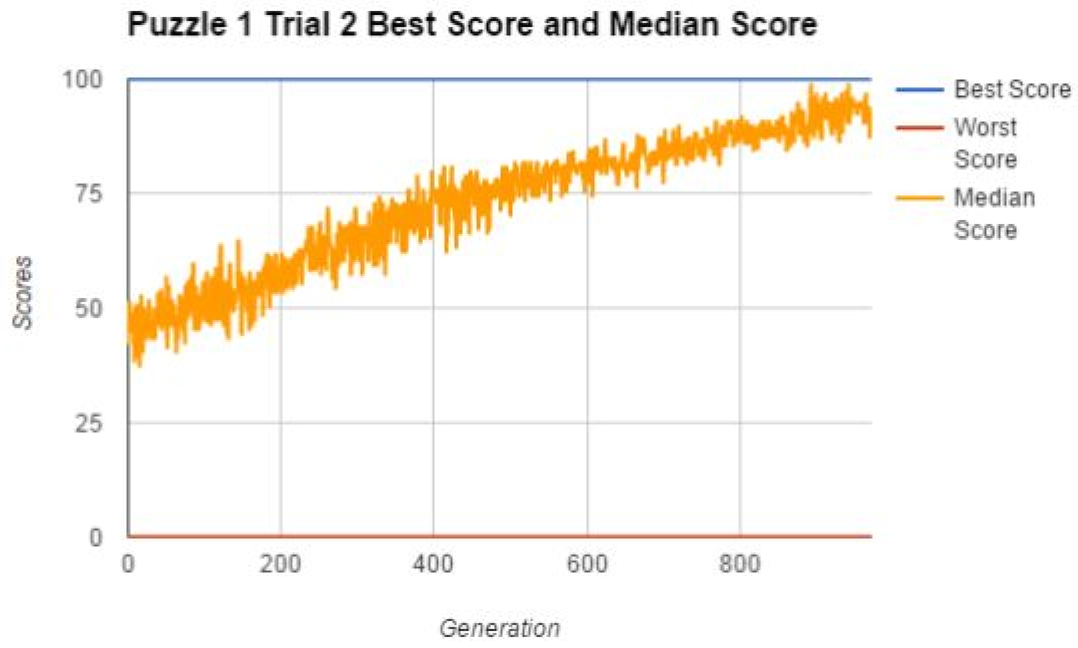
a. On the x-axis, plot the number of generations. If you want, you can sample, and only examine performance every so often (e.g., every 100 generations)

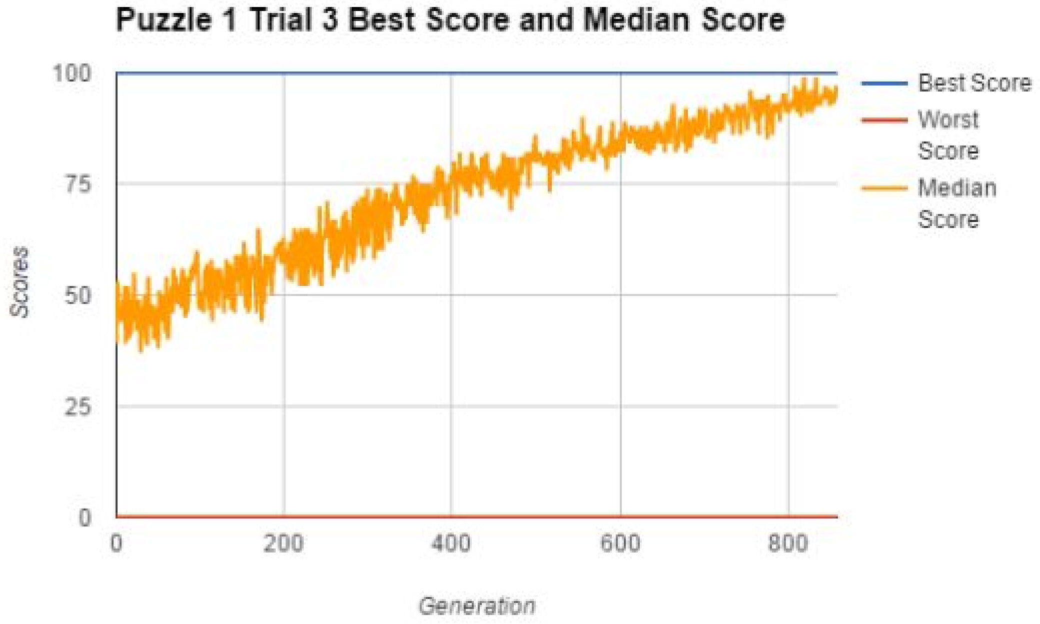
b. On the y-axis, plot performance of your algorithm. Specifically, plot the best performance of the population, the worst performance, and median (50th percentile) performance.

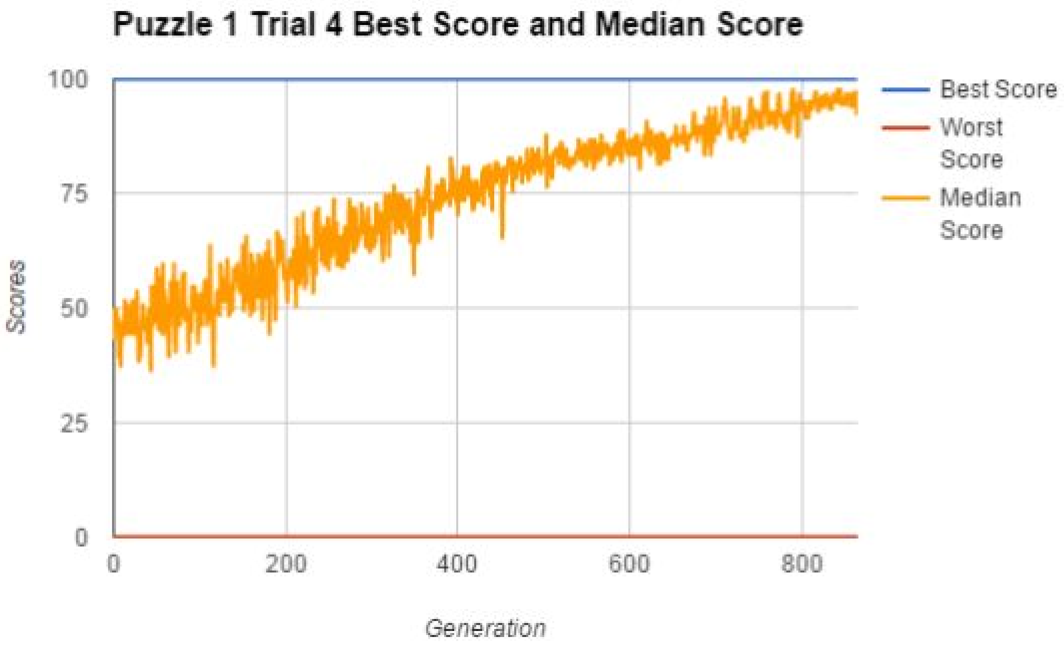
c. As GAs are non-deterministic, you should run your code 5 times on each puzzle, and report the average result on each of the metrics when generating the graphs. So you will have one graph that summarizes 5 runs of your agent on a puzzle.

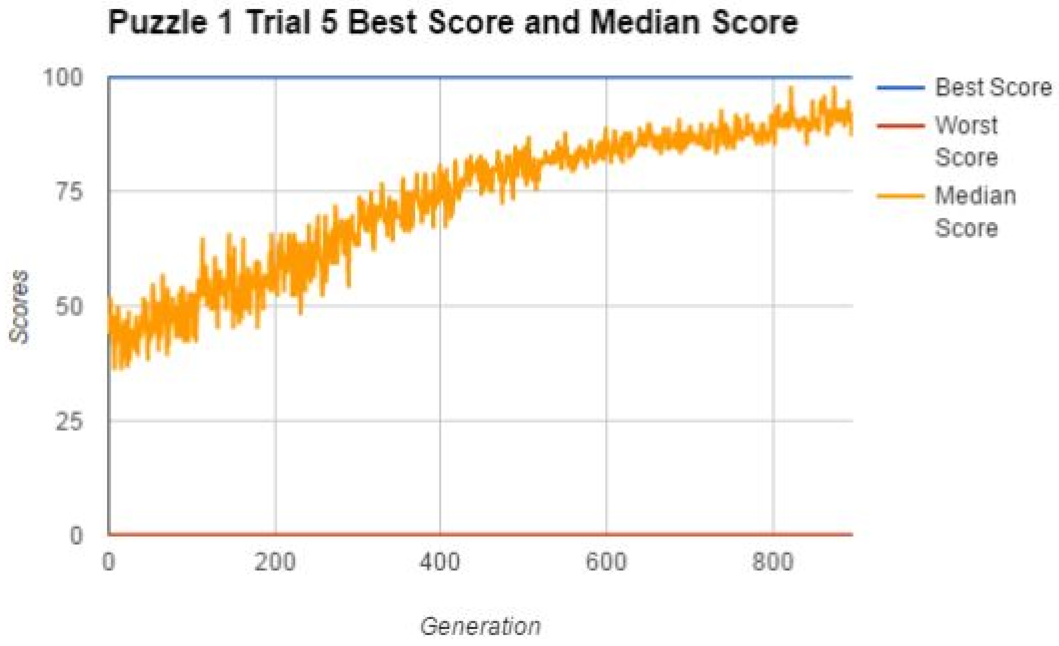
Ans:

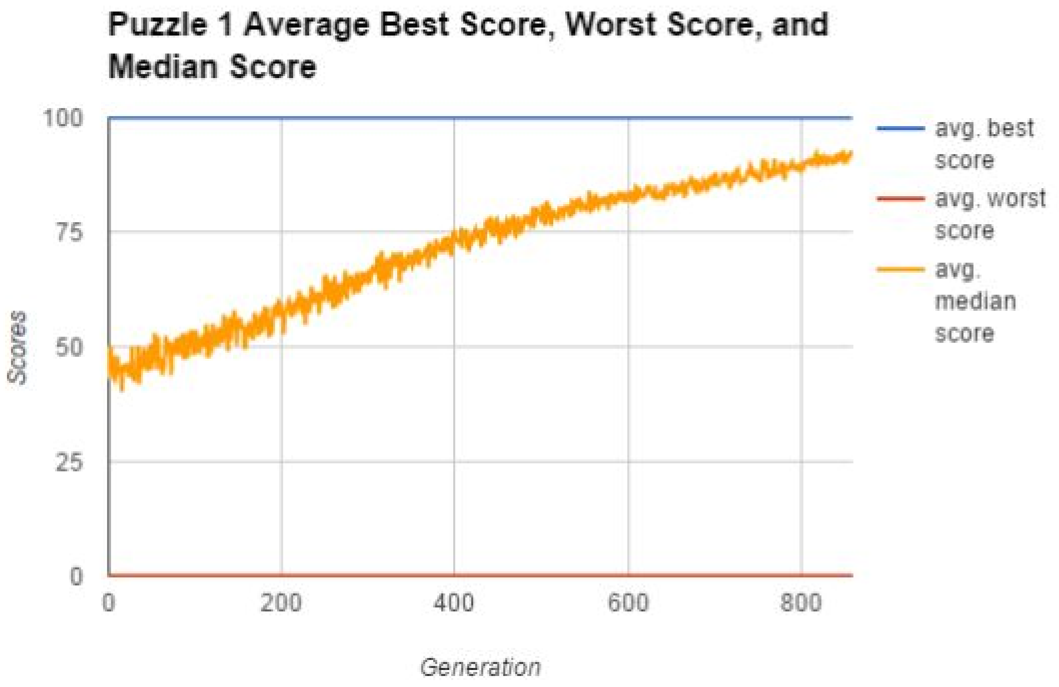



Puzzle 1 Trial 2 Best Score and Median Score
100
75
50
25
0
Scores
0
200
400
600
800
Generation
Best Score
Worst Score
Median Score

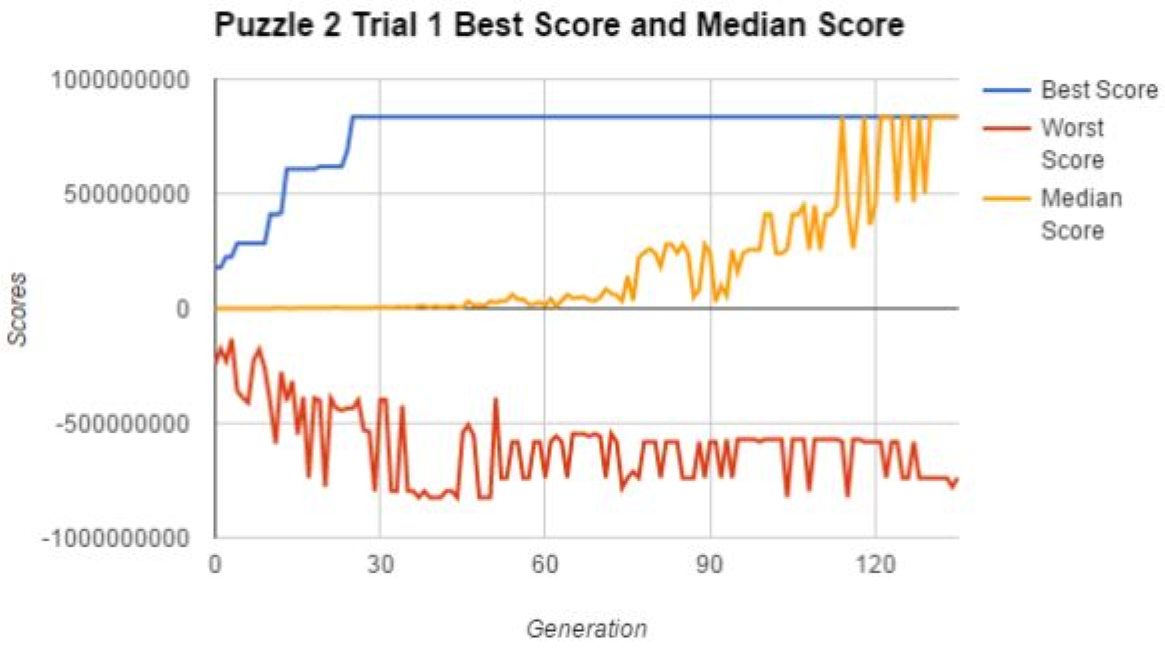

Puzzle 1 Trial 3 Best Score and Median Score
100
75
50
25
0
Scores
0
200
400
600
800
Generation
Best Score
Worst Score
Median Score


Puzzle 1 Trial 4 Best Score and Median Score

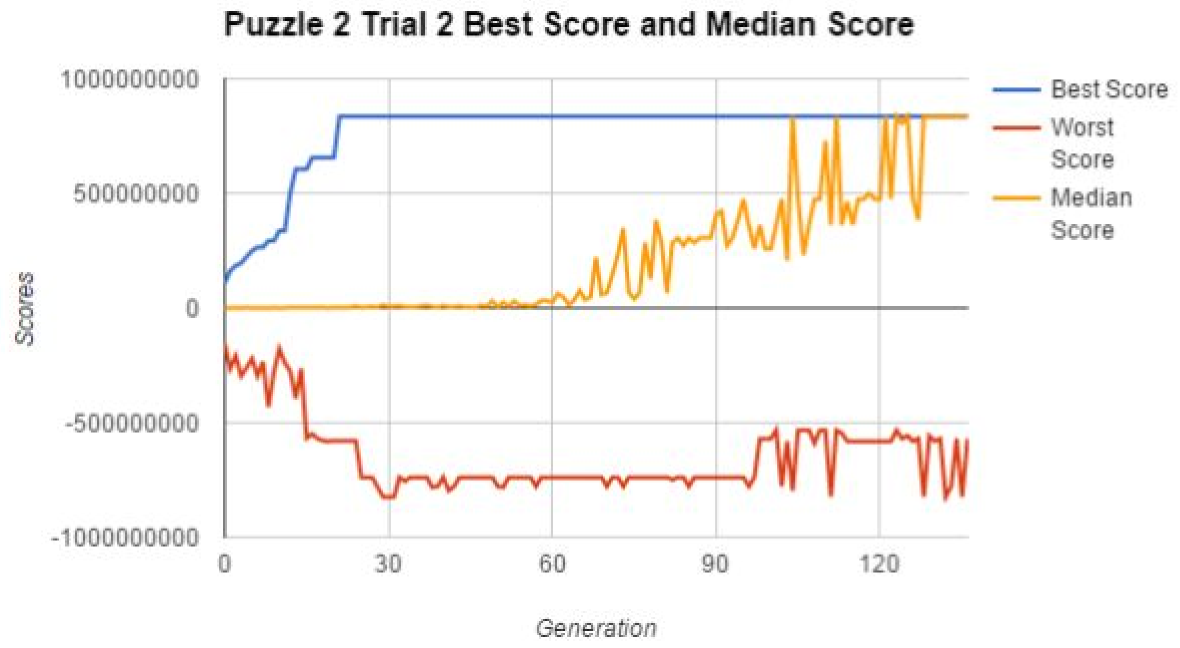

Puzzle 1 Trial 5 Best Score and Median Score

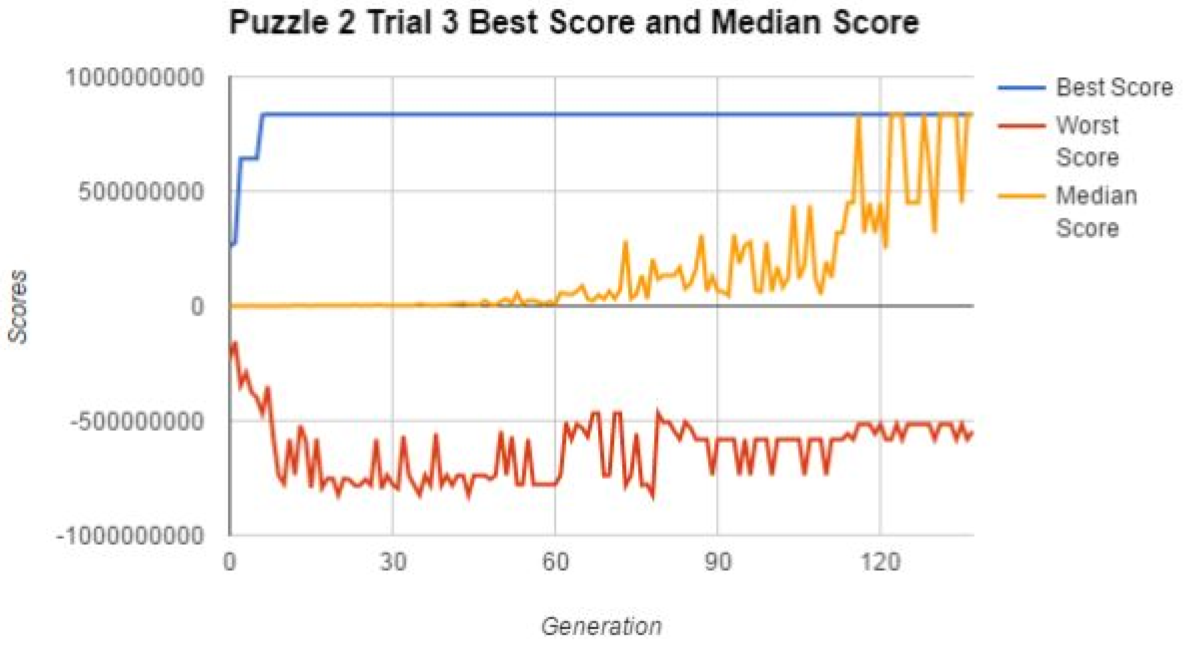


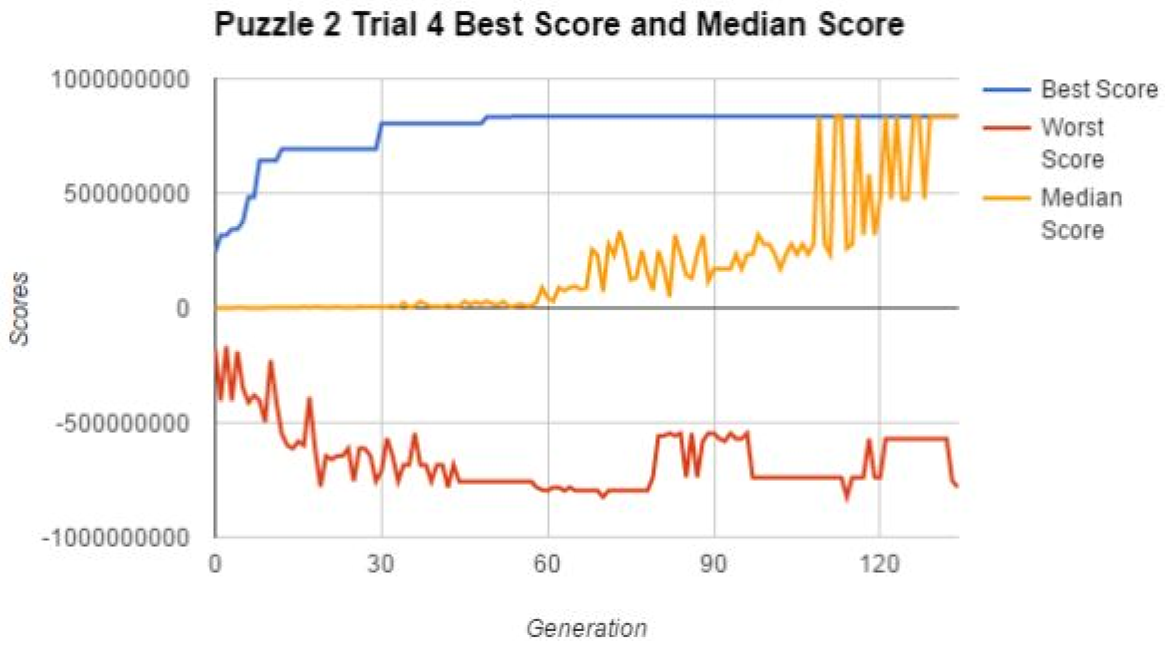
Puzzle 1 Average Best Score, Worst Score, and Median Score

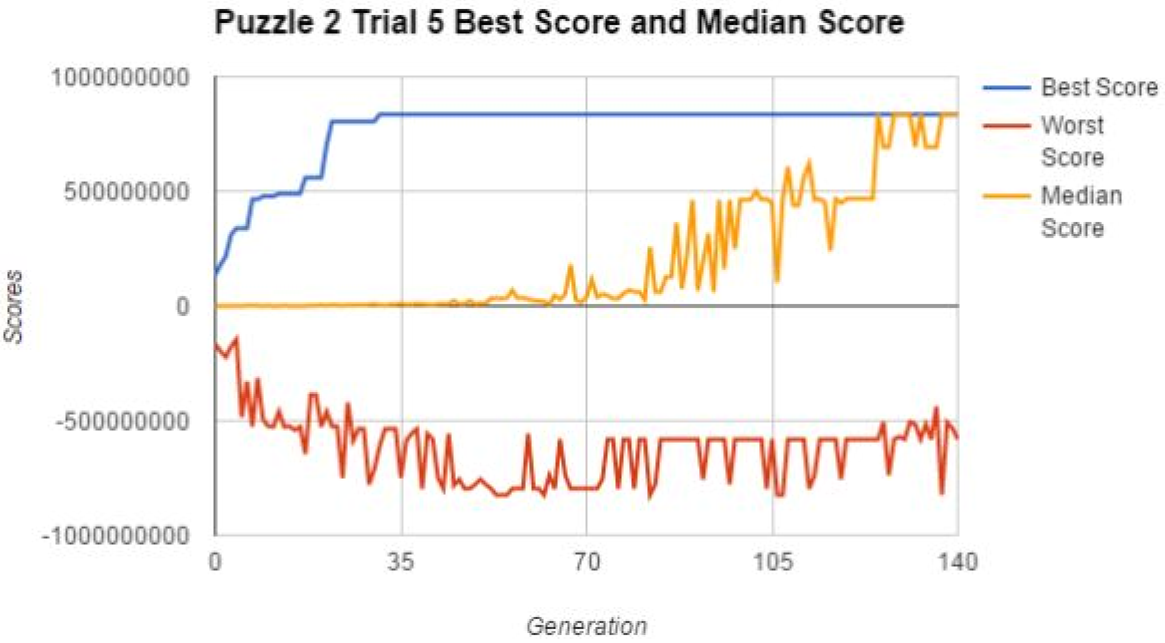


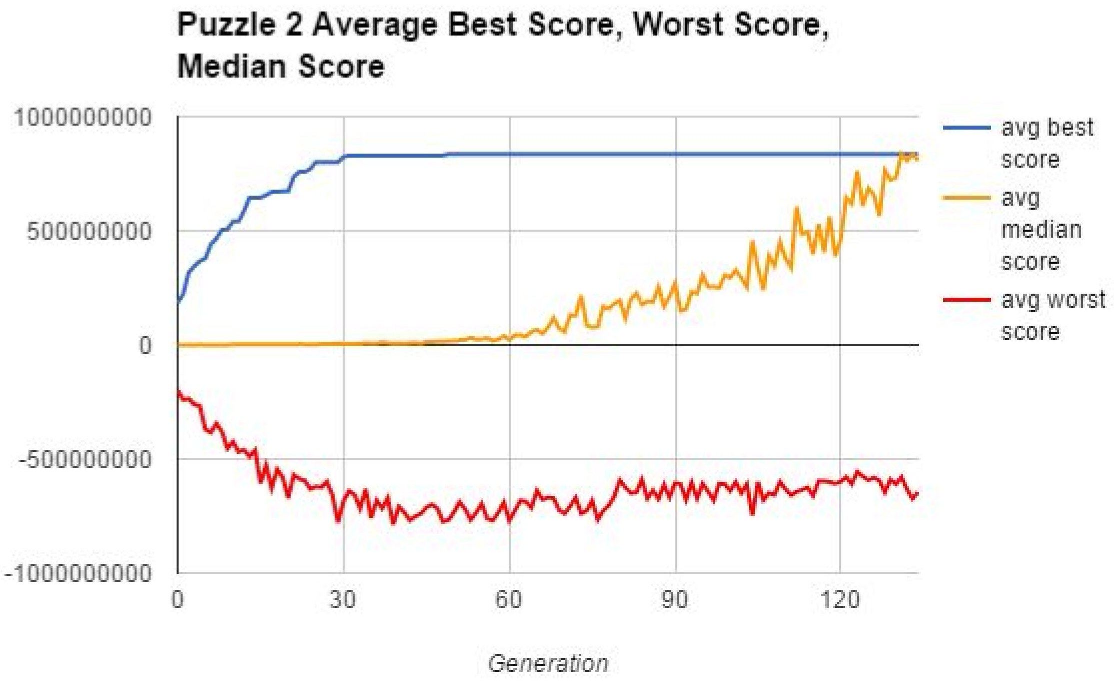
Puzzle 2 Trial 1 Best Score and Median Score

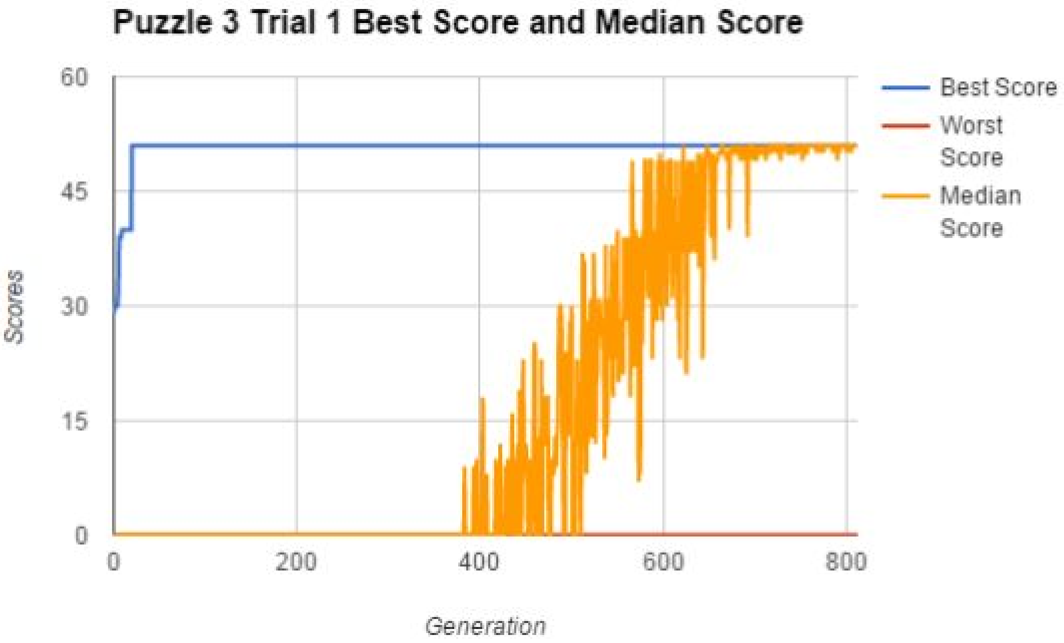

Puzzle 2 Trial 2 Best Score and Median Score
1000000000
500000000
0
-500000000
-1000000000
Scores
0
30
60
90
120
Generation
Best Score
Worst Score
Median Score


Puzzle 2 Trial 3 Best Score and Median Score
1000000000
500000000
0
-500000000
-1000000000
Scores
0
30
60
90
120
Generation
Best Score
Worst Score
Median Score

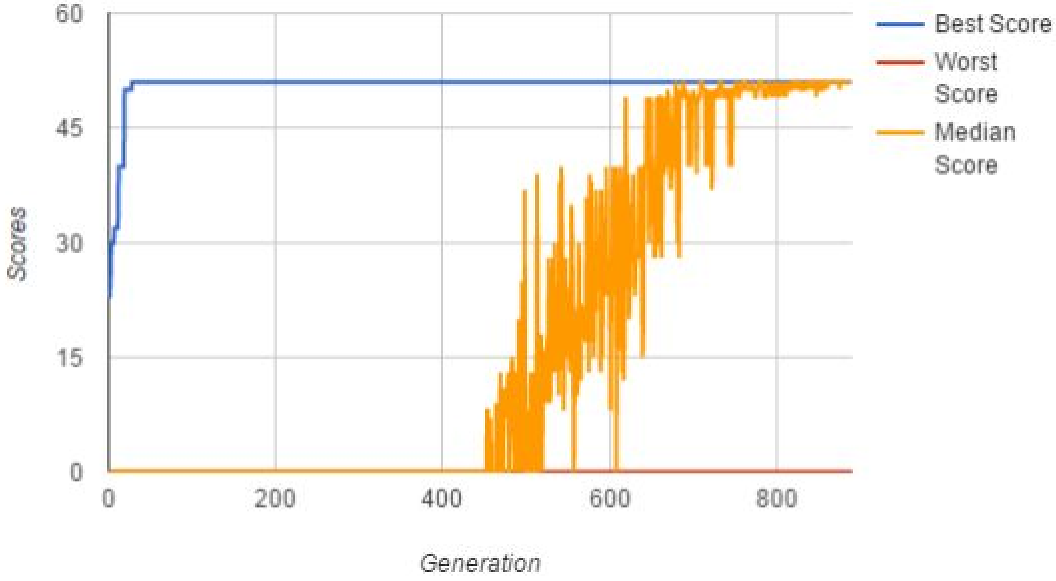

Puzzle 2 Trial 4 Best Score and Median Score
1000000000
500000000
0
-500000000
-1000000000
Scores
0
30
60
90
120
Generation
Best Score
Worst Score
Median Score


Puzzle 2 Trial 5 Best Score and Median Score
1000000000
500000000
0
-500000000
-1000000000
Scores
0
35
70
105
140
Generation
Best Score
Worst Score
Median Score

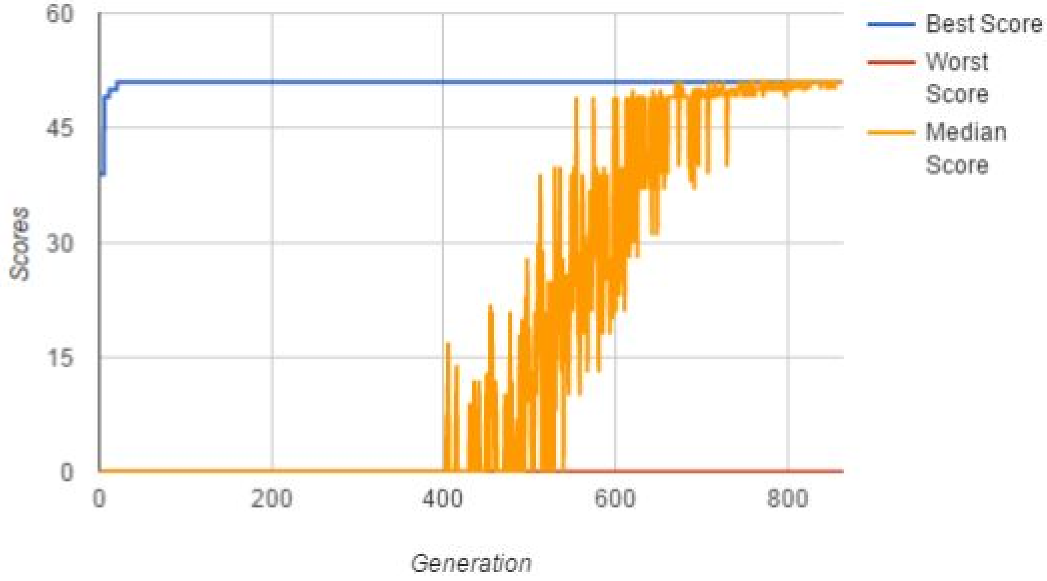

Puzzle 2 Average Best Score, Worst Score, Median Score
1000000000
500000000
0
-500000000
-1000000000
0
30
60
90
120
Generation
avg best score
avg median score
avg worst score


Puzzle 3 Trial 1 Best Score and Median Score
60
45
30
15
0
Scores
0
200
400
600
800
Generation
Best Score
Worst Score
Median Score

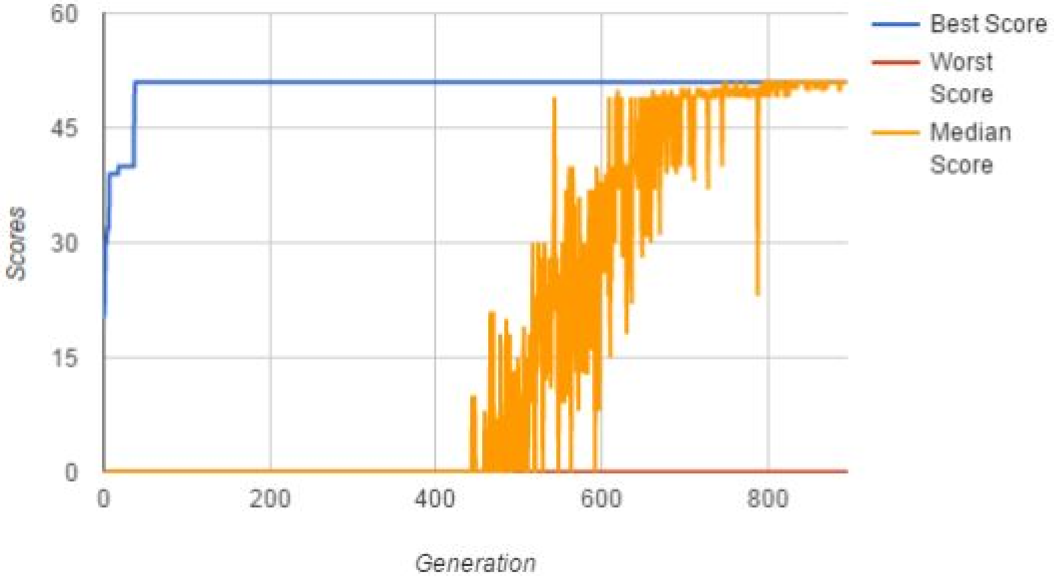
Puzzle 3 Trial 2 Best Score and Median Score



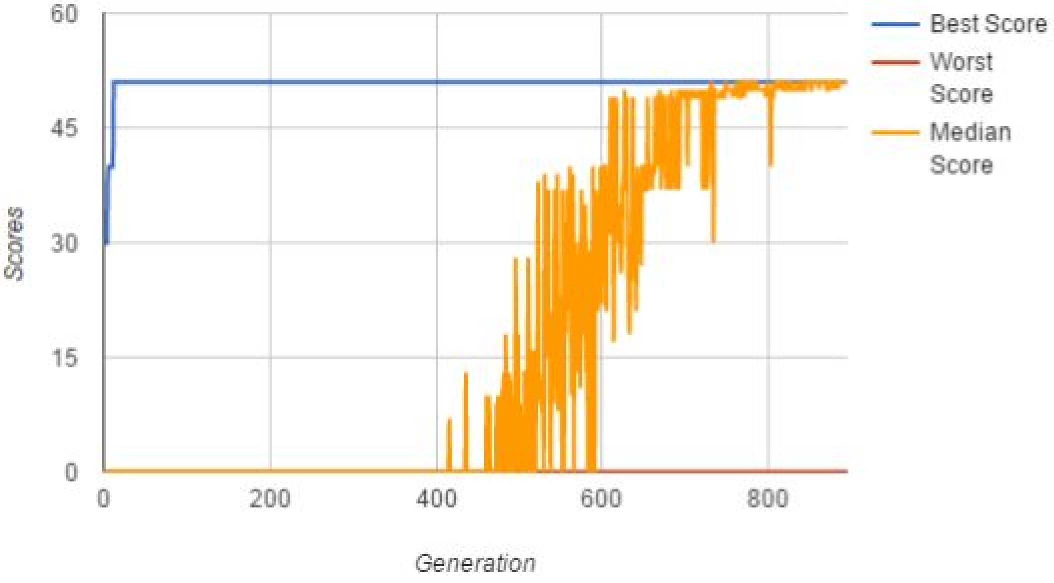
Puzzle 3 Trial 3 Best Score and Median Score

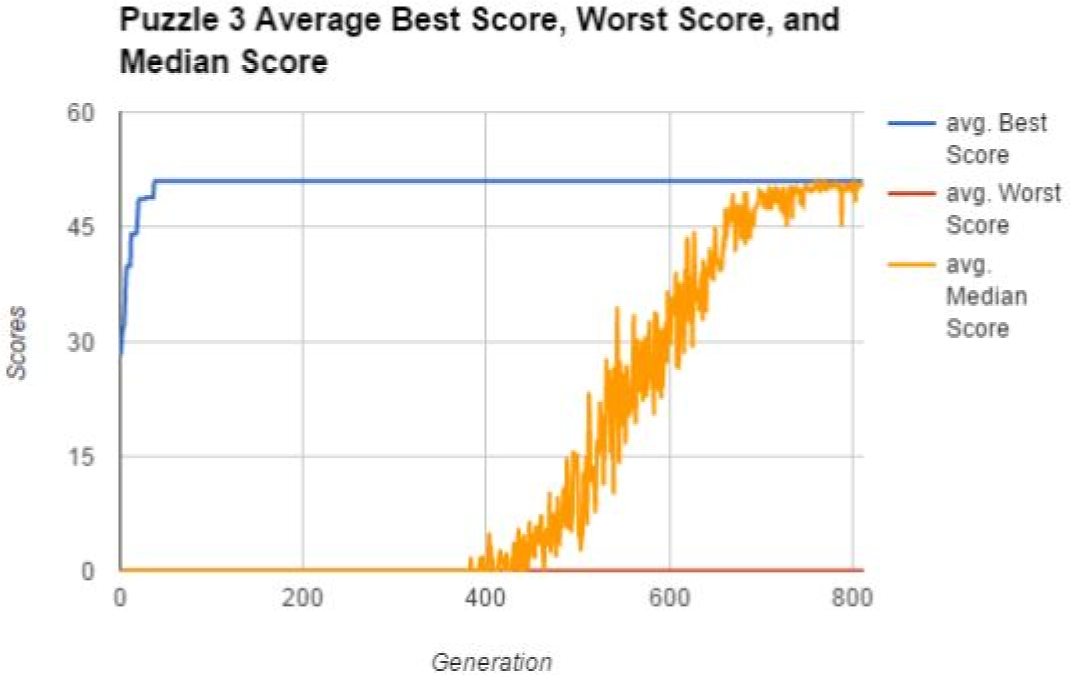


Puzzle 3 Trial 4 Best Score and Median Score



Puzzle 3 Trial 5 Best Score and Median Score




Puzzle 3 Average Best Score, Worst Score, and Median Score
Scores
60
45
30
15
0
0
200
400
600
800
Generation
avg. Best Score
avg. Worst Score
avg. Median Score

4. You must also experiment with a few simple techniques with GA and explain how they impact performance on the 3 puzzles. In addition to your baseline performance, you should generate graphs like the ones described in step #3, with and without:

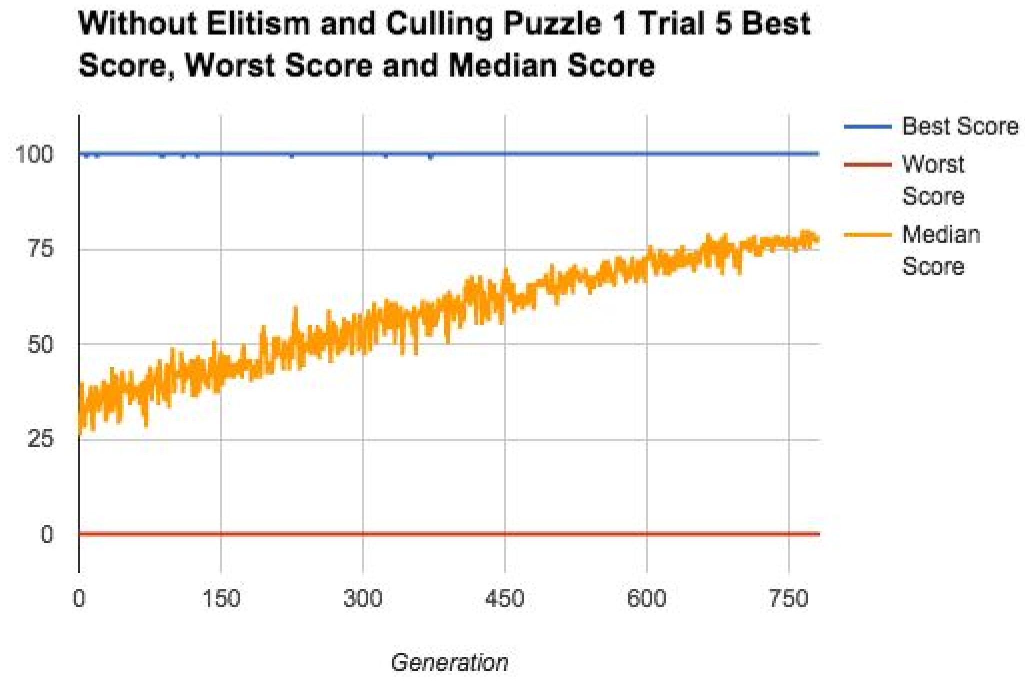
a. Elitism: experiment with retaining the top-performing members of the population on each generation.

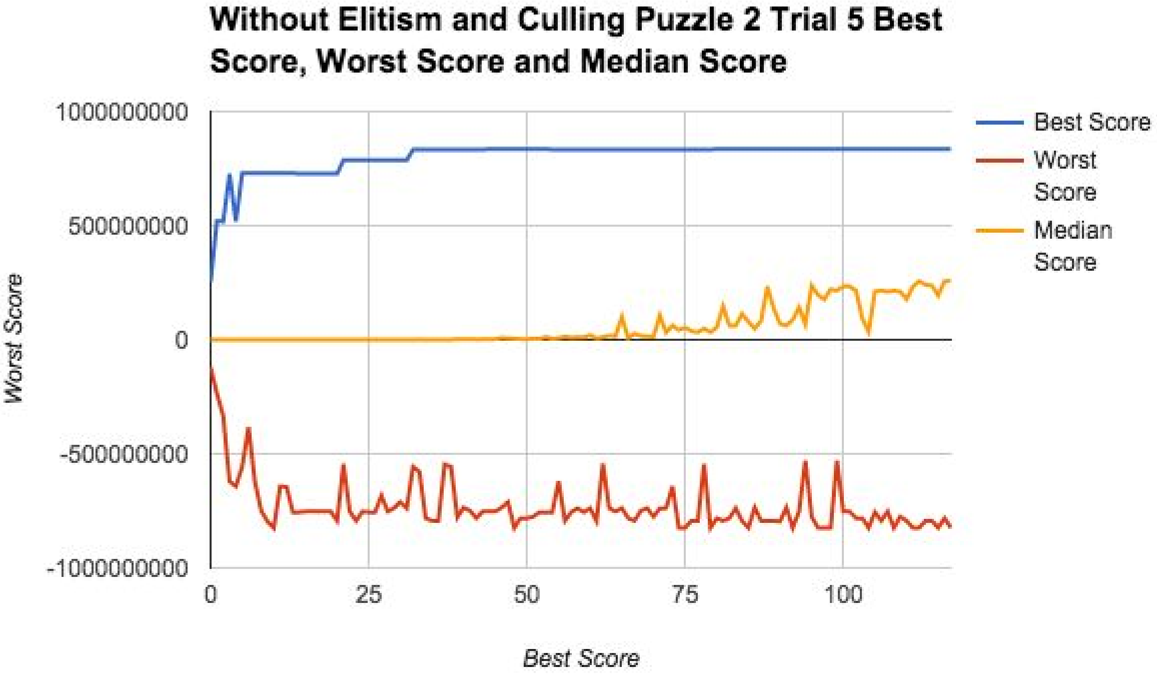
b. Culling: experiment with dropping the bottom-performing members of the population on each generation.

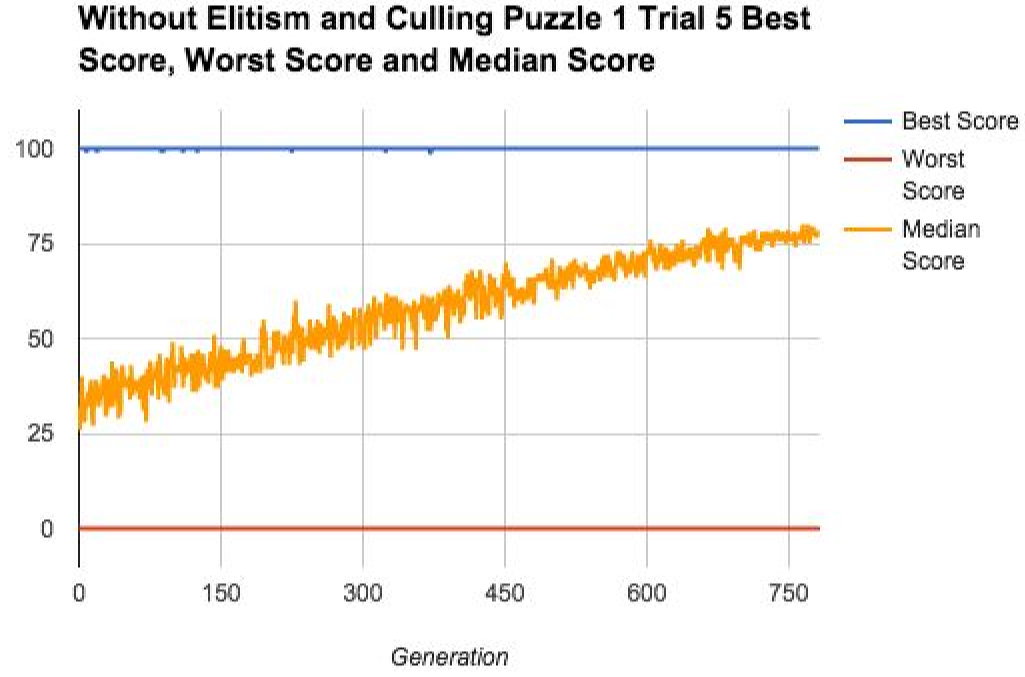
c. Explain what effect these two techniques have on the 3 metrics (best-, worst-, and median-performance)

Ans:

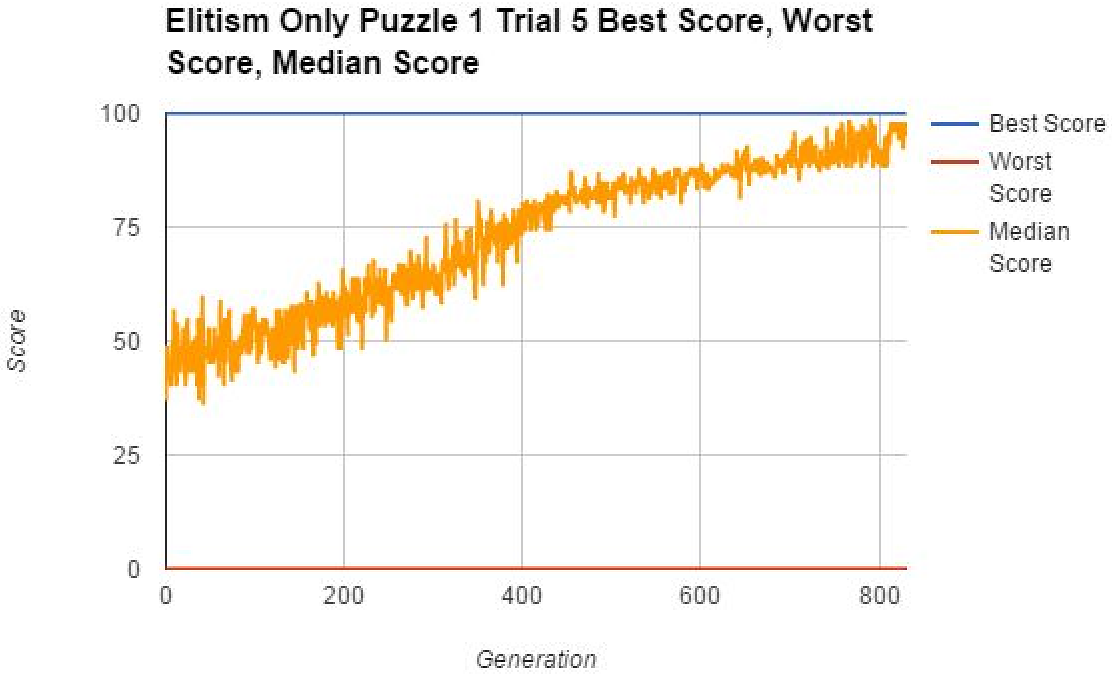
No Culling & No Elitism:

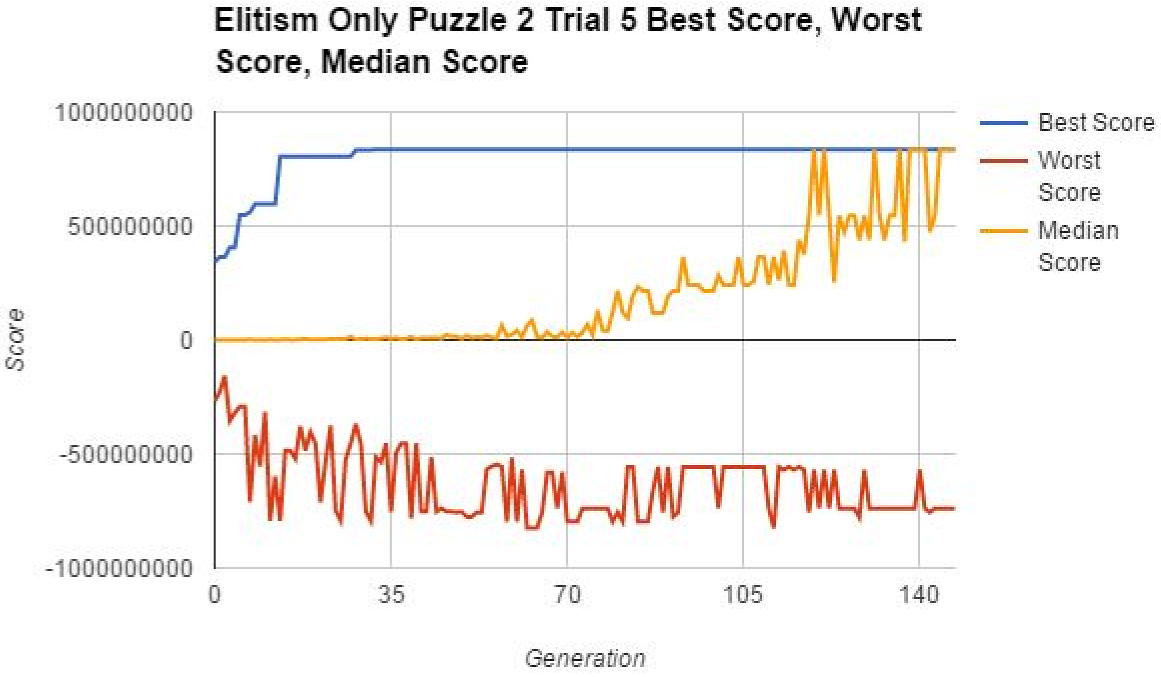


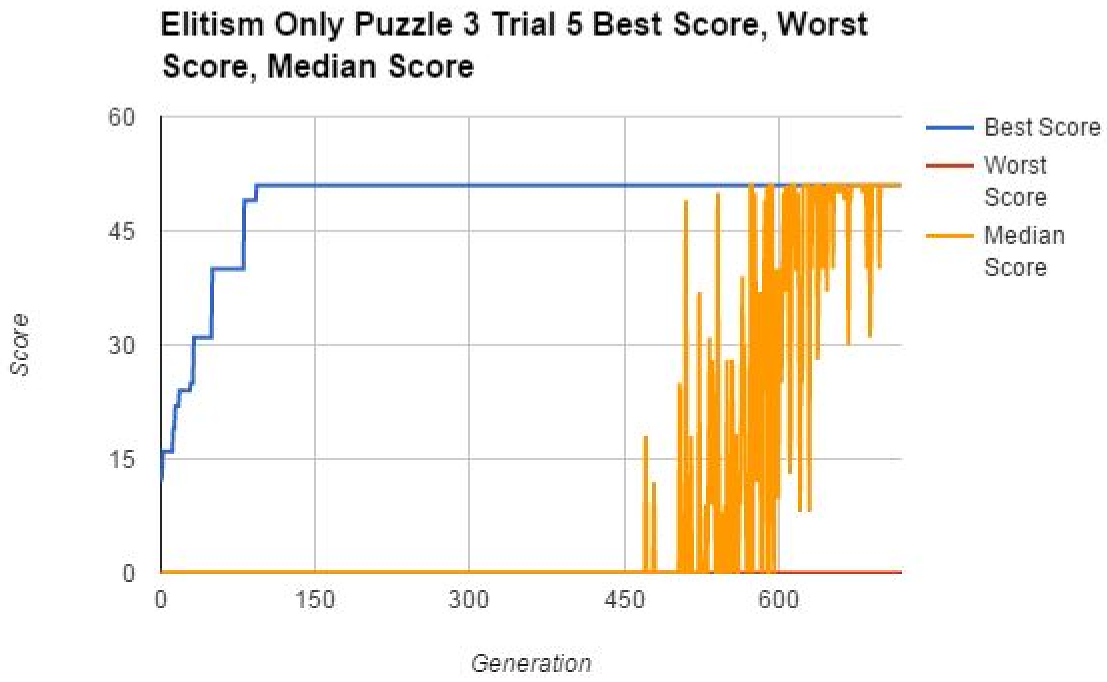

Without Elitism and Culling Puzzle 2 Trial 5 Best Score, Worst Score and Median Score
1000000000
500000000
0
-500000000
-1000000000
Worst Score
0
25
50
75
100
Best Score
Best Score
Worst Score
Median Score


Without Elitism and Culling Puzzle 1 Trial 5 Best Score, Worst Score and Median Score
100
75
50
25
0
0
150
300
450
600
750
Generation
Best Score
Worst Score
Median Score

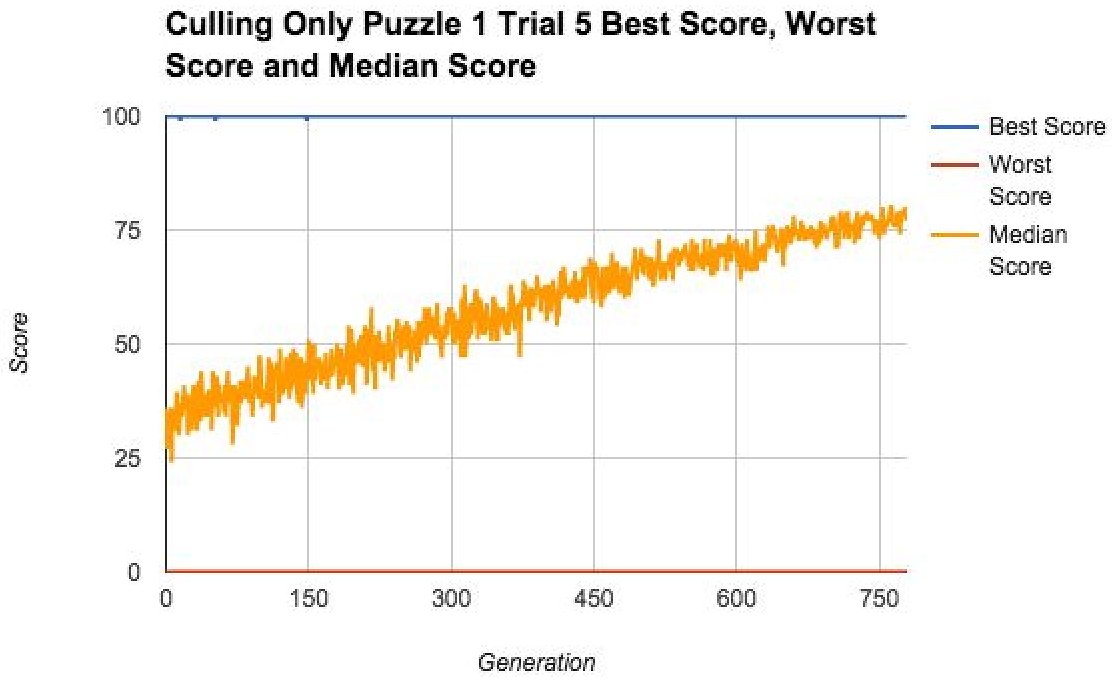
Only Elitism:

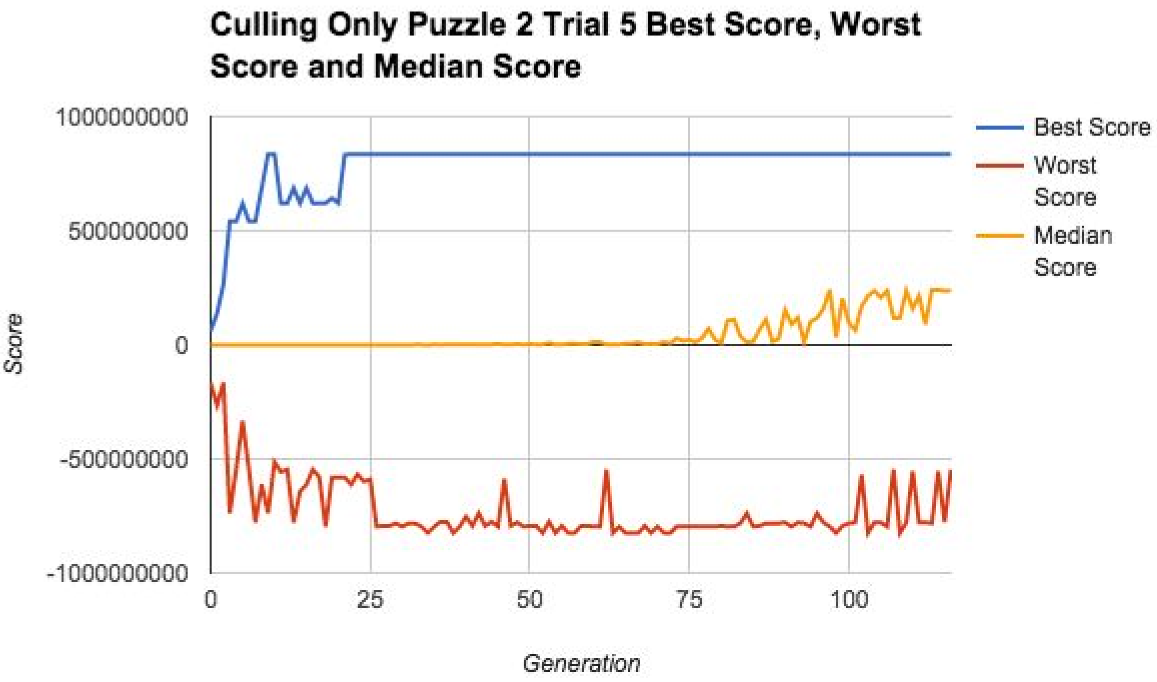


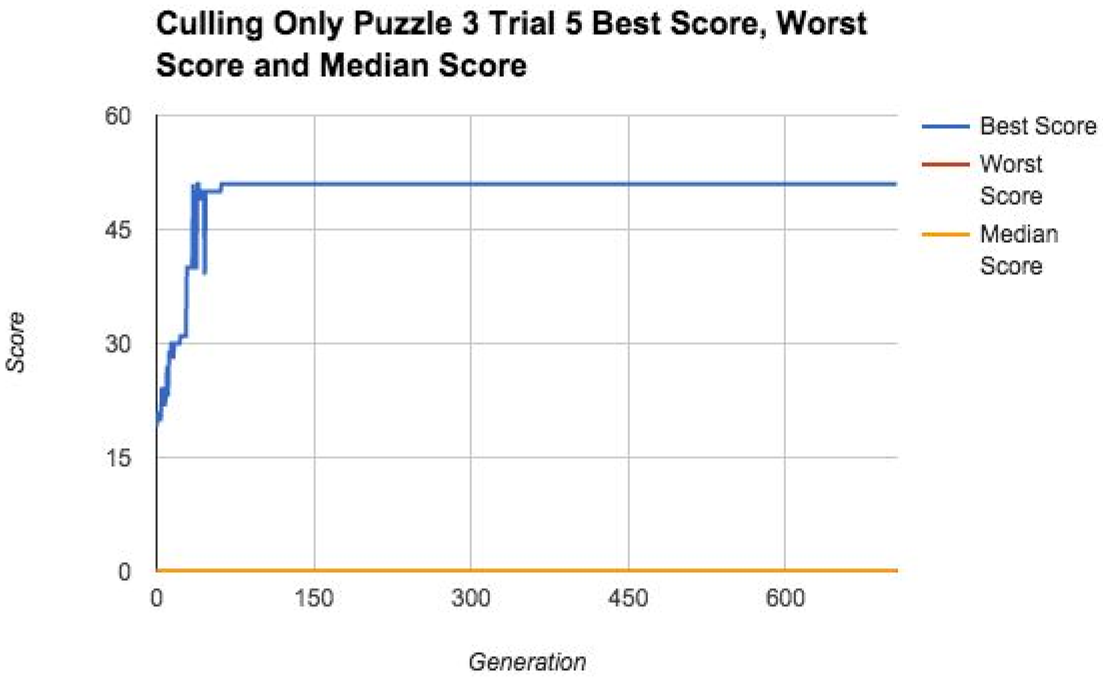




Only Culling:







Ans:

From the data, elitism seems to increase the rate at which the median score rises, and culling seems to increase the rate at which a best score is reached.