Letters and Numbers

An Investigation into Genetic Algorithms

Mattias Winsen – N10467874
Ethan Griffiths – N10467858
Connor Browne – N10486925

6 June 2021

Introduction

In one of the games in the television quiz game, "Letters and Numbers", participants are tasked with calculating a target number with a given set of six random numbers. The team has been approached with modelling this game with the intention of implementing a player of the game, using Genetic Algorithms (GA).

This was done by implementing python code which evolved the population of an expression tree, where each expression's fitness was computed by calculating the difference in the value the expression returned and the target value.

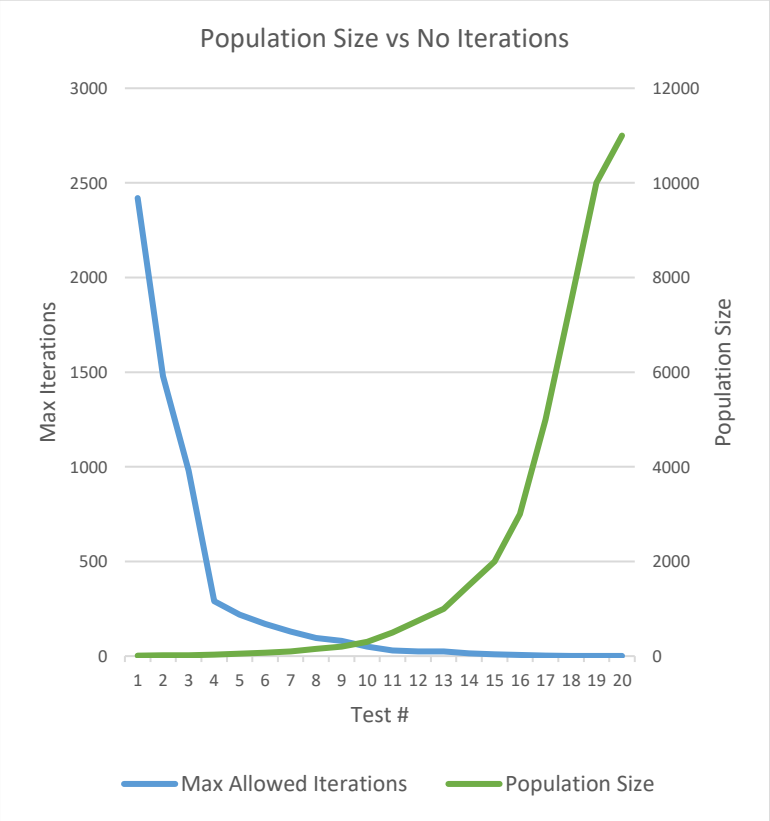
This implementation was then studied to find a suitable balance between population size, and number of iterations, within a given time.

Investigation

To investigate the effects of population size and number of iterations several simple testing algorithms were developed.

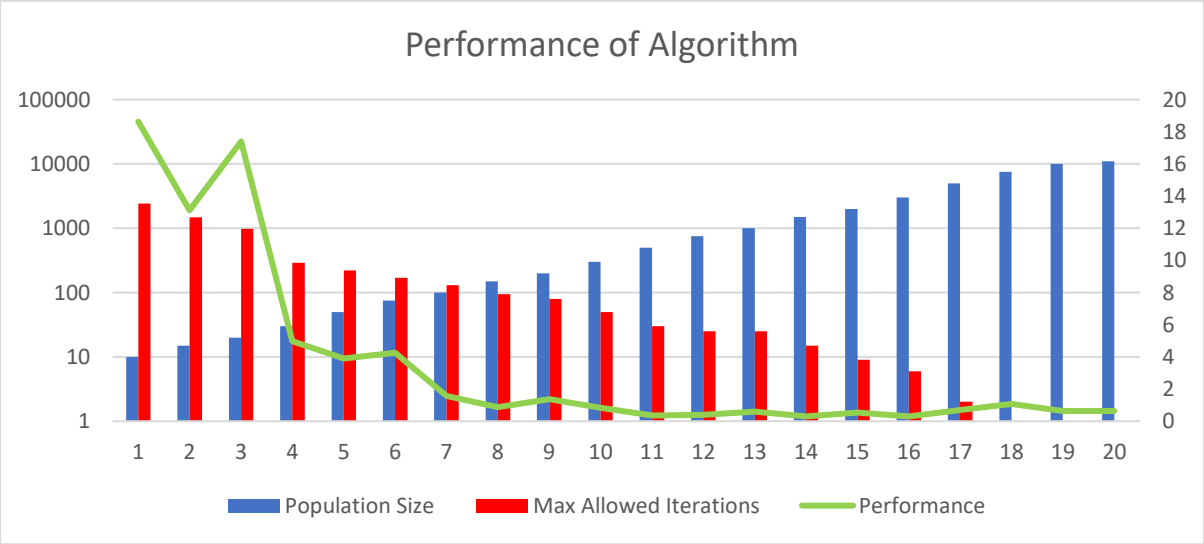
First among these, was a function to find the maximum population that can be evaluated on a particular computer in under two seconds. This was found to be 11000 for the PC used. From this function, a set of twenty population values (from 0 to the maximum population) were chosen. This set was then passed into a function which found the maximum number of iterations each population could compute in the two second time limit. This resulted in the below test data set.

Test #	*Population*	*Iterations*
1	10	2420
2	15	1480
3	20	980
4	30	290
5	50	220
6	75	170
7	100	130
8	150	95
9	200	80
10	300	50
11	500	30
12	750	25
13	1000	25
14	1500	15
15	2000	9
16	3000	6
17	5000	2
18	7500	1
19	10000	1
20	11000	1



Each pair of population size and number of iterations were then passed into the number game algorithm, to see how it performed over 30 different problems. The performance was evaluated by calculating the average difference from the algorithms calculated value, and the "perfect score" value.

Test #	*Population*	*Iterations*	*Performance*
1	10	2420	18.63
2	15	1480	13.13
3	20	980	17.4
4	30	290	4.967
5	50	220	3.9
6	75	170	4.267
7	100	130	1.567
8	150	95	0.867
9	200	80	1.367
10	300	50	0.834
11	500	30	0.36
12	750	25	0.4
13	1000	25	0.6
14	1500	15	0.3
15	2000	9	0.53
16	3000	6	0.3
17	5000	2	0.7
18	7500	1	1.067
19	10000	1	0.63
20	11000	1	0.63



Deconstructing the test results identifies several different trends. Firstly, it can be clearly identified that the process performs substantially better with an increasing population size vs an increasing number of iterations, meaning the algorithm does not need number of iterations to get an acceptable result. It can also be seen that after a certain point, increasing the population size has a very minimal effect on the program's performance.

Recommendations

The performance results found in the investigation show that there is an ideal range which generation count, and iteration count should fall between for peak algorithm performance. In this specific case (results will vary across different environments) this range is between Test #11 and Test #16. As such, it is recommended that the algorithm be run with a very low iteration count of 2 to 30 iterations and a large population count of 300 to 3000. This will result in the best possible performance from the program in while remaining under two seconds of computation time.