Symbolic Regression with Genetic Programming

Thanushan Pirapakaran
Adrian Binu
COSC 4P82
Prof: Brian Ross
Brock University
February 7th, 2024

What is Symbolic Regression?

Symbolic regression is a way to find any mathematical formulas that best match a set of data points. Each individual in the population represents a possible function to find the one that fits the data the best. This is helpful when you are dealing with complex relationships between variables and you want to understand them better. It is used in many fields like in business when you want to see future stock prices of a business.

Parameters	Settings 1	Settings 2	Settings 3	Settings 4
Pop. Size	300	300	300	300
Crossover rate	90%	90%	100%	0%
Mutation Rate	10%	10%	0%	100%
Generations	50	50	50	50
Number of Elites	2	0	2	2
Min Tree Size (init)	0	0	0	0
Max Tree Size (init)	17	17	17	17
Min Tree Size (Mut.)	0	0	0	0
Max Tree Size (Mut.)	17	17	17	17
Tournament Size	3	3	3	3

Table I: Parameters of four separate experiments

Fitness Function

The program reads in x-values through a text file and passes through the fitness function. The fitness function assesses the fitness of individuals within the population of a given generation. Each individual represents a mathematical expression. The function evaluates these expressions to generate predicted y values, which are then compared to the actual y values. The fitness is determined by finding the difference between the predicted y values generated by the individual expressions and the corresponding actual y values. Specifically, it calculates the difference between the root value produced by the individual expressions and the actual y values. Thus, returning a value that is used as a fitness to guide the evolution process of the genetic program.

Experiments

For all experiments done with each setting, the dependent variable was the fitness solution. Thus, a lower fitness resulting in a steady downward decline (in generations vs. fitness plots), would suggest a better resulting experiment.

Setting 1:

The first setting illustrated in Figure I, has a changed independent variable of 2 elitism.

Setting 2:

Similarly, we tested the effects of elitism by experimenting with the same crossover rate, mutation rate, etc., but with an elitism of 0. Comparing setting 1 and setting 2, then allowed us to visually see the differences, and come to a conclusion. This setting's results can be found in Figure II.

Setting 3:

On the other hand, we tested two extreme cases, to interpret the effects of crossover vs. mutation. Thus, in Figure III, we tested the impacts of a 100% crossover rate vs. a 0% mutation rate.

Setting 4:

Furthermore, we tested the effects of a 0% crossover rate vs. a 100% mutation rate, to inspect the impact of mutation (illustrated in Figure IV).

Comparisons

In our comparative analysis, we conducted 10 runs per setting to evaluate the impacts of elitism, mutation, and crossover. Our aim was to determine the best settings for an average genetic program. By examining the resulting average fitness, average best fitness, and average tree size graphs, we can visually select a setting that yields the best overall GP performance, highest fitness, and minimal bloat.

Results

In terms of elitsm, we found that there were little to no effects on performance and fitness, when comparing setting 1 and 2. When comparing the data in Figure I to the data in Figure II, it is pretty much identical. Not to mention that they resulted in the same best fitness (when using the same random seed). As a result, we have come to conclusion that elitism has little to no effects on overall performance. While our comparison focused on 0 vs. 2 elitism, a larger gap would have result in a bigger difference. Nevertheless, a higher elitism value essentially transforms the system into another hill climbing algorithm.

When comparing the two extreme cases in setting 3 and 4, there is a lot of randomness. The best fitness plots in Figure III and Figure IV are the shape of an optimally performing GP system. However, as they can be chalked up to elitism, we can neglect those graphs when comparing the effects of crossover

vs. mutation. Comparing the average fitness (of Figure III and IV) is where the effects of mutation and crossover can be found. In particular, the average fitness of 100% mutation has no resemblance to a normal performing GP system. Instead, it completely randomizes the evolutionary process. Thus, a low mutation (like 10% mutation rate used in setting 1 and 2) is the best choice to ensure the GP system doesn't become a random search algorithm. Moreover, a higher crossover rate is better than a lower one, as it is less random and surprisingly causes the least amount of bloat (as depicted in Figure III average tree size plot). Nonetheless, a 100% crossover rate is still too high as it may add too much randomness to the GP system.

In fact, settings 1 and 2 follow all the requirements of an optimal GP system above. While, also providing the best overal fitness from all 40 runs: 3.89038E-33

.

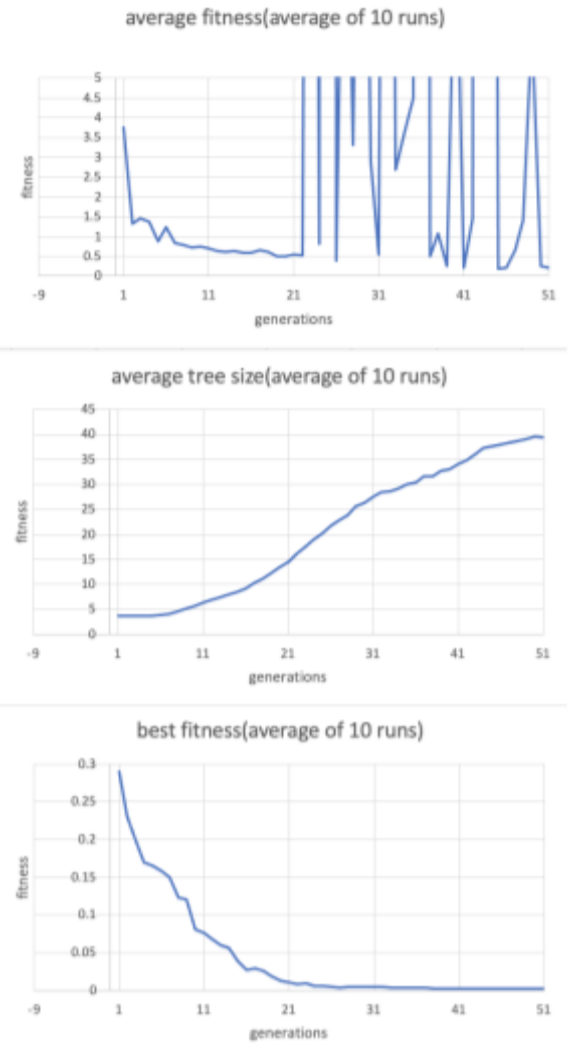
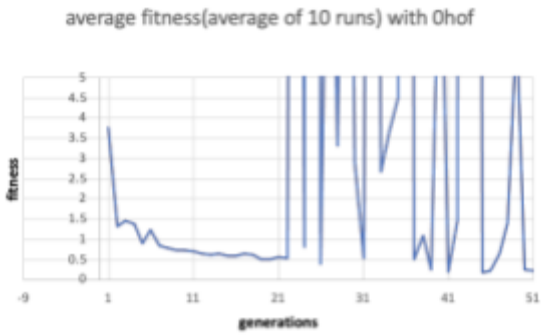
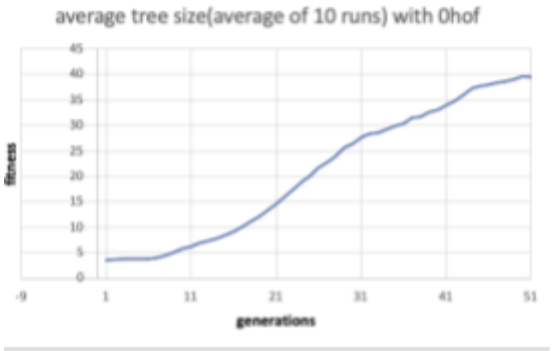


Figure I: 90% Crossover, 10% Mutation, 2 Elitism

Conclusion

All in all, the criteria to find the best settings for a GP system includes a low mutation rate, a high crossover rate, any amount of elitism (as long as its not extremely high), and ensures that there isn't too much randomness. Thus, the best setting we have found for the purposes of symbolic regression is setting 1 or 2.





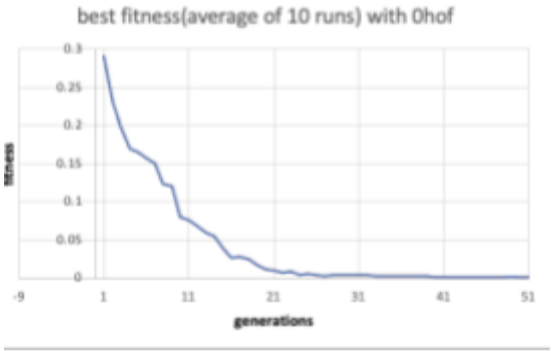
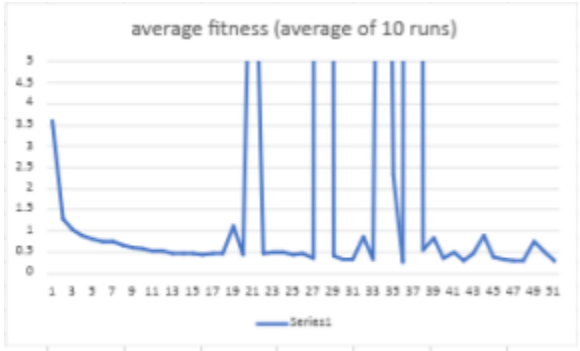
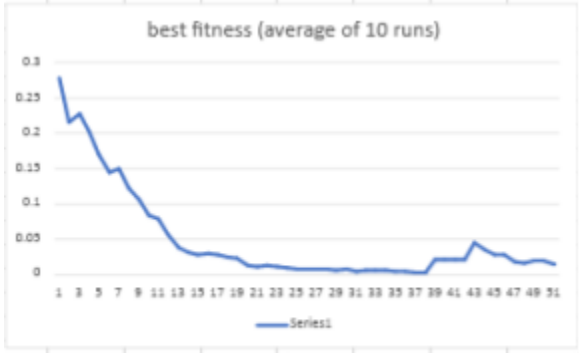


Figure II: 90% Crossover, 10% Mutation, 0 Elitism





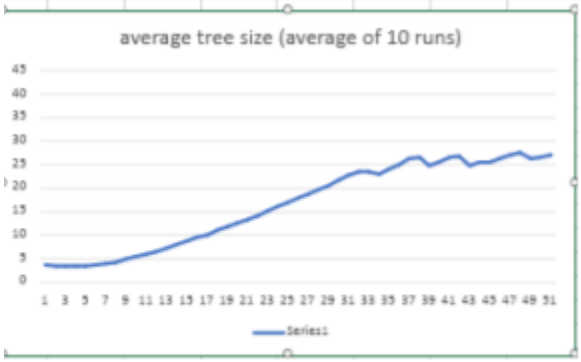


Figure III: 100% Crossover, 0% Mutation, 2 Elitism

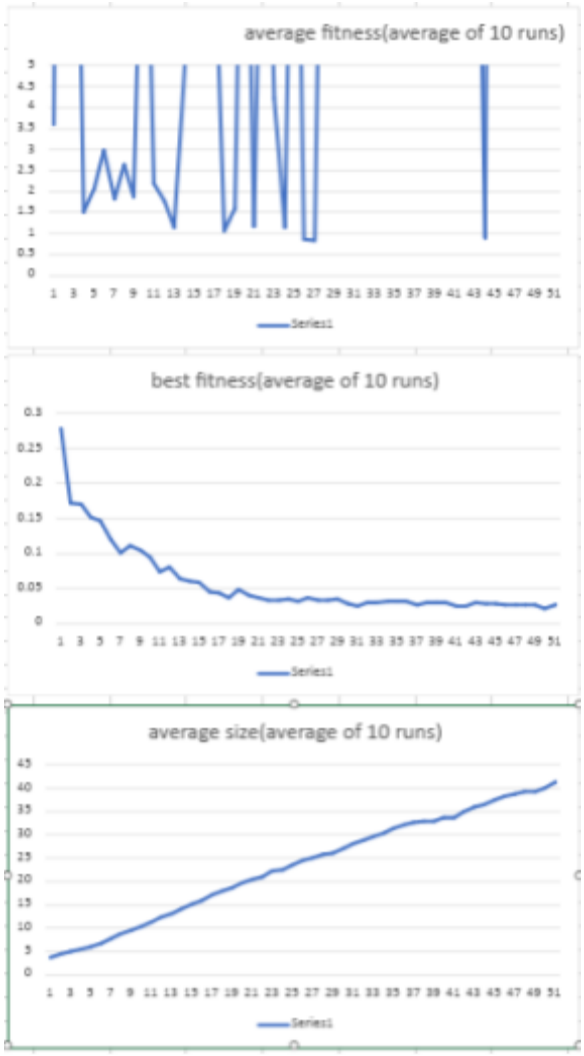


Figure IV: 0% Crossover, 100% Mutation, 2 Elitism

Function and Terminal Set

Function	Arity	Example
ADD	2	x + y
SUB	2	x - y
MUL	2	x * y
Protected_DIV	2	x / y (1 if y = 0)
NEG	1	-x
COS	1	cos(x)
SIN	1	sin(x)