GeneticPID White Paper

The representations can be shown with [t_r, t_s, m_p], and the final values after running the algorithm is [0.70, 15.86, 34.08]. The parameters are being represented in real values, and each gene (parameter) has a 25% chance to mutate to any value which is two decimal points specified by the given bounds. Each allele (three parameter combination) has a 60% chance to experience crossover and uses uniform crossover.

The fitness function is the inverse of the ISE (1 / ISE) so that the algorithm searches for a higher fitness factor. What this means is that the algorithm optimizes for a higher fitness factor, because if it were simply ISE and not the inverse of the ISE, then it would have to optimize for the lower factor (which would be considered more fit here).

The graph of the fitness over generations with the parameters provided is present in Figure 1.

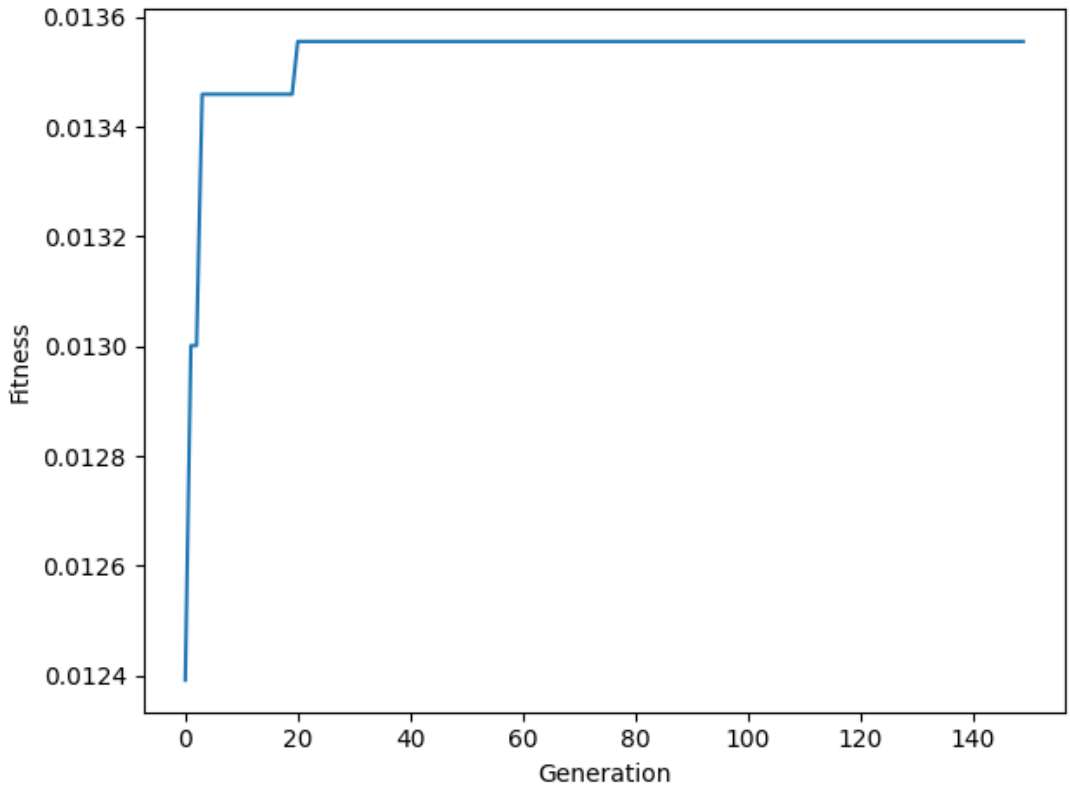

Figure 1. Fitness (1 / ISE) over 150 generations.

Modifying from the base graph present in Figure 1, it is possible to run the algorithm with various numbers of generations. This is present in Figure 2.

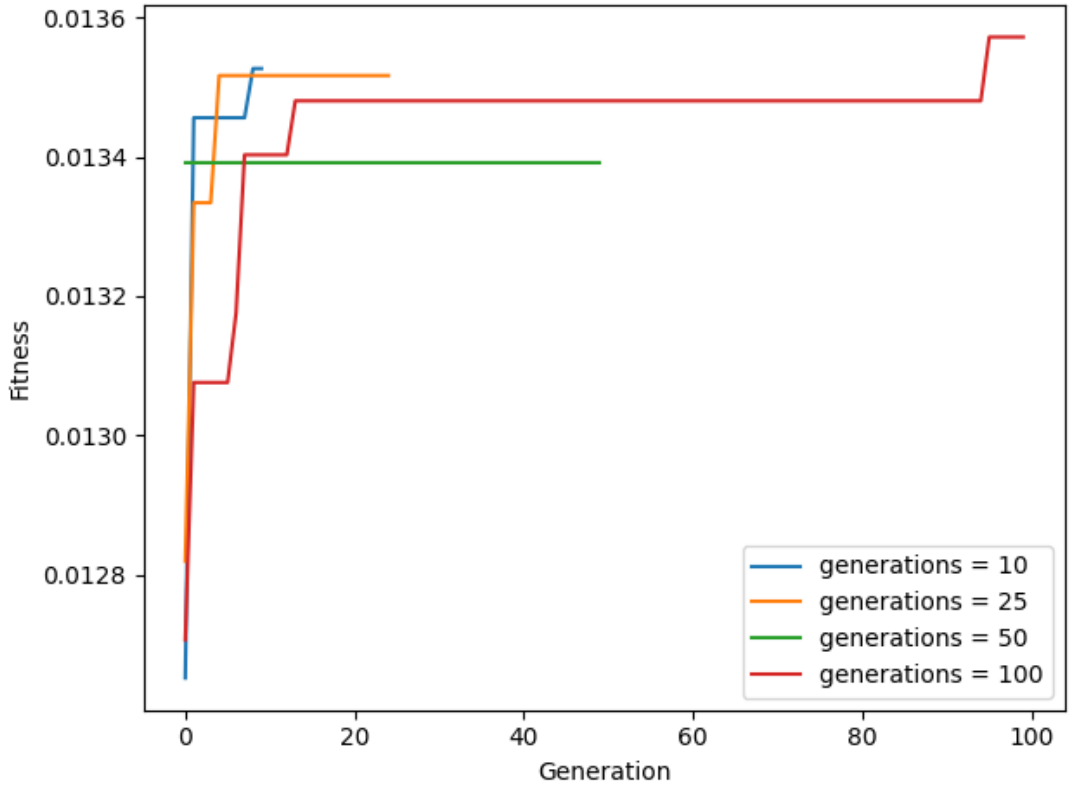

Figure 2. Fitness (1 / ISE) over generations.

Therefore, as the number of generations increase, the solution tends to get better. This is because there are more iterations to allow for crossover and mutations to occur, resulting in a higher probability of finding a better solution. However, the most drastic changes occur in the first few generations, but the more generations there are, generally, the better the solution will become. Nevertheless, this is probabilistic, meaning that generally having more generations is better, but it is also possible, albeit less probable, that the same test with fewer generations finds a better solution than one with many. This is why the orange line in Figure 2 has a better fitness than the green line.

The next variable to be modified is population count, which will be modified various times with a generation count of 50 because most changes occur in the first 50 generations, and with this smaller scale, the differences will be more visible. This is present in Figure 3.

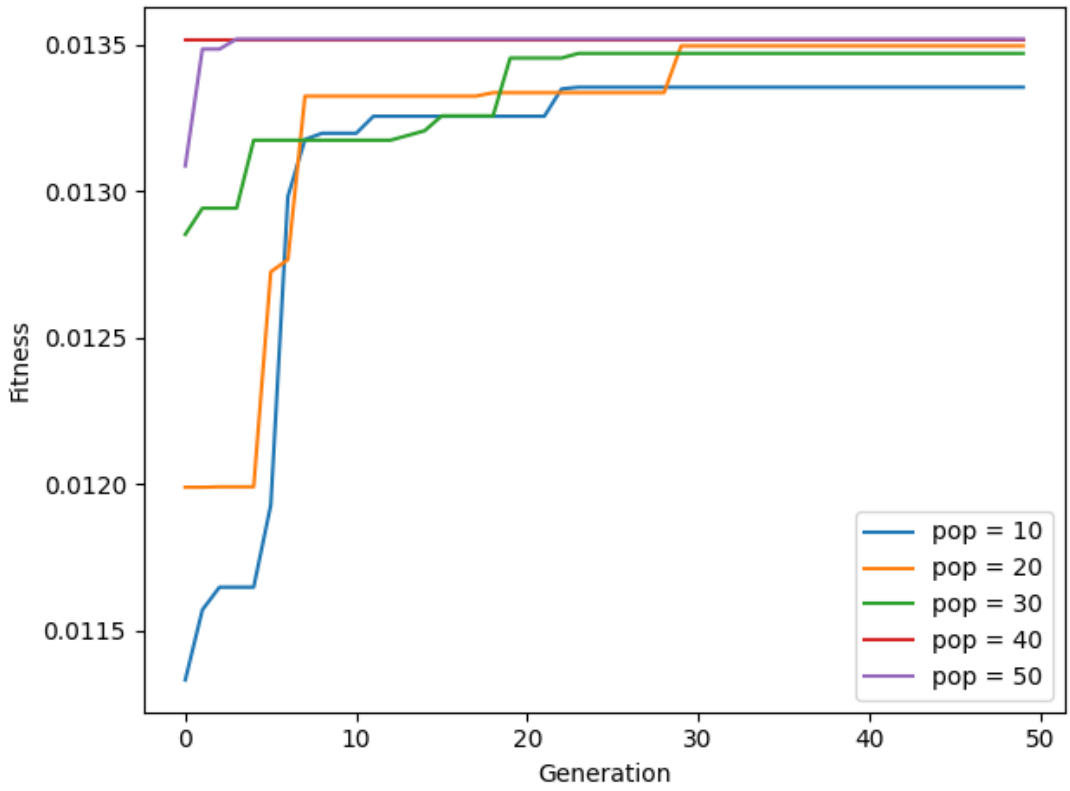

Figure 3. Fitness (1 / ISE) with various population sizes.

This shows that as population size increases, the number of possibilities increases, meaning that each generation will have a better fitness, in general. This means that a better fitness will be reached each generation, and that a better fitness will be reached overall also. However, it is probabilistic, so generally a higher population is better, but it is not always the case.

The next alterable variable is the crossover probability. The crossover probability affects how likely uniform crossover is to occur. The default value is 60%, but various modifications are shown in Figure 4. The crossover technique used is uniform crossover, meaning that if a crossover occurs, each parameter will have a coin toss which decides whether a crossover will occur at that parameter or not.

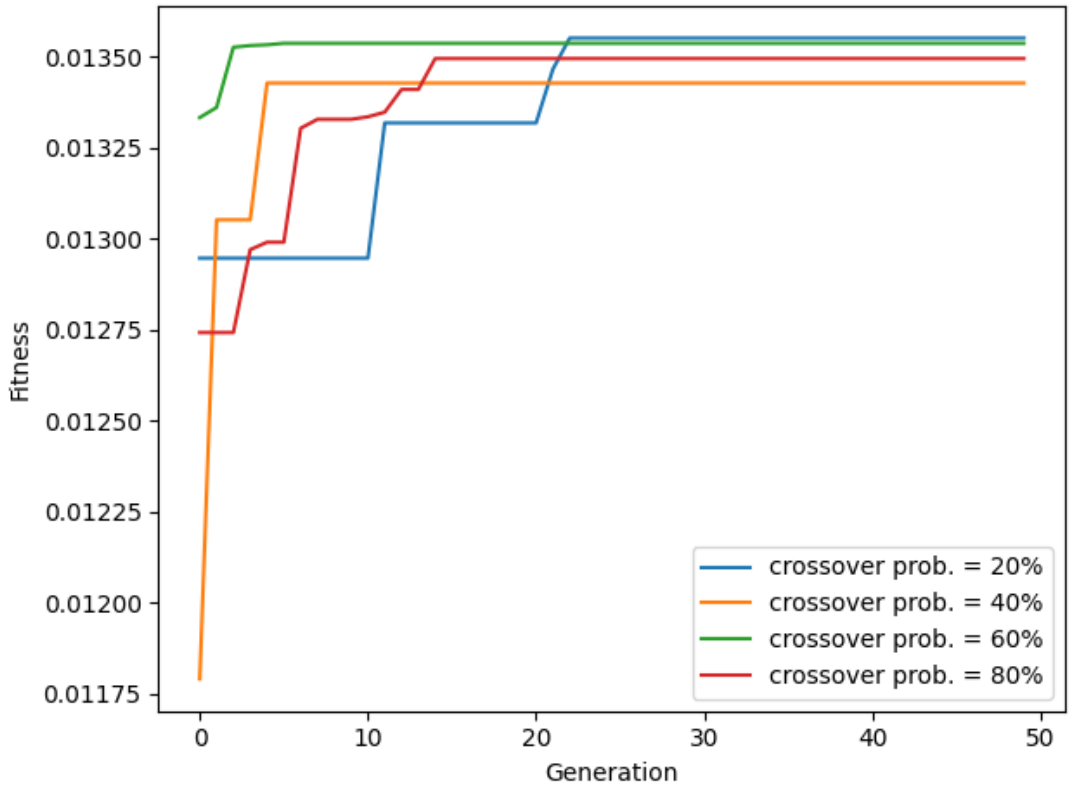

Figure 4. Fitness (1 / ISE) with various crossover probabilities.

From this graph, the best crossover line is in green, which is 60%, because it reaches a higher fitness value earlier on. Blue reaches a higher fitness after a while, but it is likely due to luck. Indeed, this is because if crossover is too likely to occur, it will explore too much, preventing the ideal fitness from being reached, and if it is too unlikely to crossover, it will not explore enough, preventing better solutions from being reached.

The final alterable variable is the mutation probability, which dictates how likely any parameter is to mutate. The default value used thus far has been 25%. This is present in Figure 5.

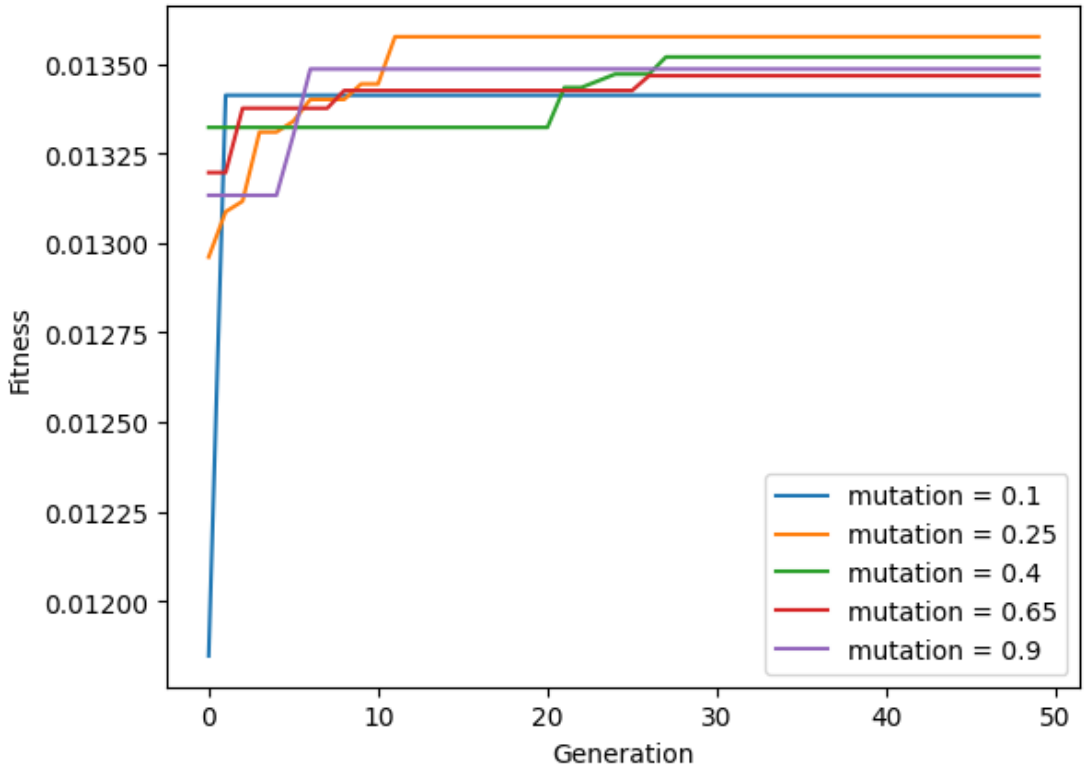

Figure 5. Fitness (1 / ISE) with various mutation probabilities.

From this graph, the best line is the orange one, which has a 25% chance of mutation on each parameter. This is because if the mutation probability is too low, then not enough children will change to explore the search space; however, if it is too high, there is not enough exploitation of a good solution.