What is Genetic Algorithms?

An algorithm is a set of rules used to solve a problem. In Genetic Algorithms, most of these problems are answered with genetics. Candidates with possible answers are tested to see if their answer is close to the correct answer. If the answer is not found, then the population is allowed to breed and children are created to try and replace their parents in the old population. The Population is then cut down through natural selection where the strongest of the population survive. This process of testing, breeding, and natural selection is repeated until an answer to the problem is found.

Testing the candidates:

In my implementation, each candidate can be scored with how many clauses the candidate makes true within the expression. Each literal is mapped to a value within the candidate. For example, if the candidate had a value of “1100” with the literals of a, b, c, and d. The literal a and b would both be 1 while c and d are 0. With the expression (a) & (b) & (!c) & (!d), the candidate's fitness score would be 1. If the value was “1000” using the same expression, the fitness score would be .75. The equation to find the fitness score would be this: number of clauses correctly answered divided by the number of clauses there are. Any candidate who receives a 1 fitness is a solution to the expression.

The Miracle of Breeding:

If a solution to the expression is not immediately found, there needs to be a change in the population. Plugging in random values could possibly get the correct equation, but the Algorithm would take much longer to determine a solution. This is where the genetic part comes into Genetic Algorithms. Two candidates of the population will create two children which will be stored into a new population. These children will inherit the back half of one of their parents and their front half will be an inverse of the back half. Once all candidates create children, the candidates from the new population will go through mutation. With chance, the each bit can possibly be flipped to a new bit value. An example of a mutated candidate would be 111111 changing to 111110. The last bit, through random chance, was mutated and changed from a 1 bit value to a 0 bit value. In my implementation, i put these newly mutated candidates on the final population list.

Natural Selection:

Now here is where we pull everything together. In my implementation, I take each member of the old population and compare it with each member of the new one. If a member of the old population has a higher fitness score than one on the new population, the old member takes it’s place in the final population. The population will then be set to the final population, effectively “killing” the old members with weaker scores. This is done to weed out candidates who have weaker fitness scores. Done over time, the fitness score of the population will rise gradually until a solution to the expression the population is trying to solve is found.