Genetic Algorithm

Miguel de la Cruz Cabello
Computer Science Department
North Central College
Naperville, Illinois
mdelacruzcabello@noctrl.edu

Even Myren Nybø
Computer Science Department
North Central College
Naperville, Illinois
emnybo@noctrl.edu

Braedon Fyock
ComputerScience Department
North Central College
Naperville, Illinois
bjfyock@noctrl.edu

Abstract

Genetic algorithms are heuristic optimization methods that, among other applications, can be used to find the value that maximizes or minimizes a function. The individuals of a population reproduce by generating new offspring, whose characteristics are a combination of the parents' characteristics (plus certain mutations). Out of all of them, only the best individuals survive and can reproduce again, thus passing their characteristics to the following generations. In this project, we present a version of a genetic algorithm that finds the shortest path that connects four nodes chosen by the user from a given graph file.

I. Population

In genetic algorithms, the term chromosome refers to each of the possible solutions to the problem. Each chromosome represents a possible combination of values (in our exercise, the chromosomes represent a path connecting the nodes in the graph). To represent these chromosomes, we initialize a population. **From the article retrieved**, performance depends more on the population size than the algorithm's number of iterations. Therefore, we decided **to input a large number that initializes the population to improve performance** ().

II. Fitness

Each chromosome in the population must be evaluated to see how good it is as a solution to the problem. A good heuristic implementation is needed for estimating the cost. In our project, we have a set of nodes we want to connect. For that reason, we want to **maximize the number of nodes a chromosome has that are equal to the nodes in our set**. To do so, we give each chromosome a cost in the hundreds, meaning that if it contains 1 node we want to connect, it will receive a score of 100. The highest score a chromosome could get then it would be 400. On the other hand, we want to **display the shortest path containing our set of nodes**. Here the cost of each path represents its length. Lastly, we create a **scores function** (our fitness function) that **combines the length and the nodes' scores of a chromosome**, selecting the chromosome with the closest score to 400 (for example, it would pick a score of 405 over 396 because that means there 4 nodes we want to connect in the 405 scores, while there are only 3 nodes in the 396 score).

III. Selection

In a genetic algorithm, the selection is a set of rules that choose the parents of the next generation. These parents will reproduce and generate new offspring**. In our project, we select one parent that will reproduce.** The probability that a chromosome gets selected is proportional to its fitness score. This means that the chromosome with the highest number of nodes and whose score is closest to 400, will be the parent to reproduce. If a chromosome does not contain any of the nodes from our initial set of nodes, it will not make it to the next generation.

IV. Crossover

The objective at this stage is to generate new chromosomes from the strongest chromosome that combine the characteristics of the best chromosome. Just as shown in the PowerPoint slides, the best-selected chromosome will cross with the other chromosomes in the population. **The strategy we used in the assignment was crossing over from a single point**. **The position of the point is randomly selected to act as a cutting point.** The parent is divided into two parts and the halves are exchanged with all the other chromosomes in the population. As a result, we end up with a new generation made that doubles the size of the initial one, as two new chromosomes are generated from each crossover. Then, another selection occurs to decide which chromosomes pass to the mutation.

V. Mutation

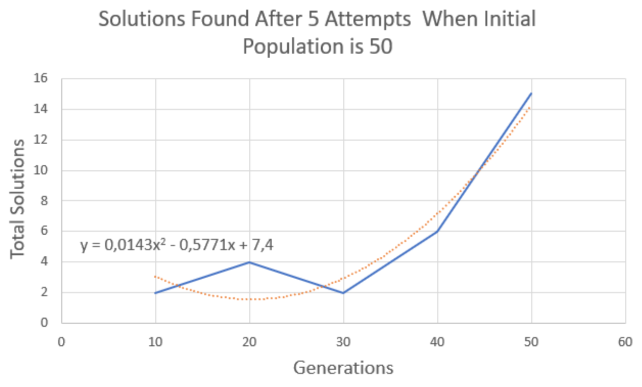
After selecting the best new chromosomes from the offspring, a mutation process takes place. **Each gene in the chromosome can be modified with a probability p. We chose to use a swap mutation, where 2 positions in the chromosome are interchanged randomly.** However, when the algorithm was being tested, we found out that the mutation was breaking the rule of the graph, as there were times when swapping the nodes broke the path. This resulted in a debug error as it was hitting a local maximum. Having a mutation that broke the rules, we had a chance of hitting a local maximum instead of having an optimal solution.

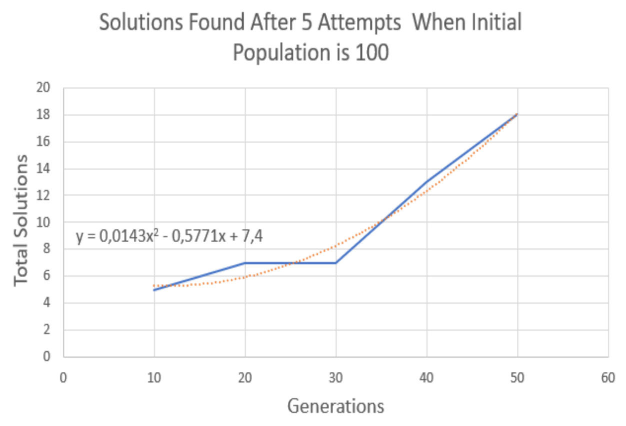
VI. SHRINK

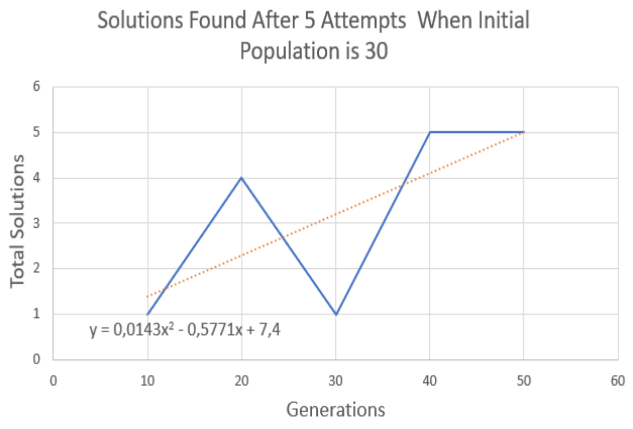
When an iteration in the loop occurs, the **shrink function is called to cut the unwanted nodes once all the nodes from the initial set are visited.** This chromosome (already shrunk) will be added to a final solutions list. When the last iteration finishes, the best chromosome from that list will be selected, meaning it is the most optimal path.

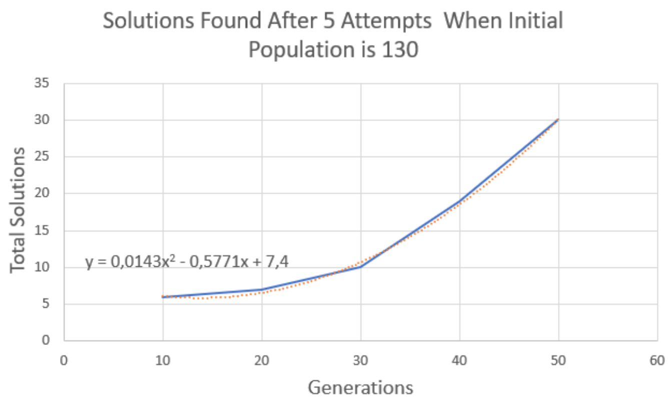
VII. EXPERIMENTAL RESULTS

Different trials on the algorithm were performed. We wanted to **determine how varying the population size affects the chances of finding a solution with a specific number of generations.** We used different population sizes (30, 50, 100, 130) and five generation limits (10, 20, 30, 40, 50). We did five attempts on each generation limit ad added all the solutions found by the algorithm. **Despite, not having an efficient mutation function, the results reveal what other studies have shown**. **Performance in finding more solutions increases when the population size and number of generations get larger**. As mentioned before, these graphs represent the number of times a solution is found after five attempts in a generation limits.









VIII. CONCLUSION

As shown in the experiment's results, it can be concluded that more solutions can be found when the initial population and the generation limit are larger. The fact of not having an efficient mutation function reduces the performance but overall, we are satisfied with the results found and the work displayed.