Genetic Algorithms – Graph Miner

Arelly Ragland
Computer Science Department
North Central College
Naperville
aragland@noctrl.edu

***Abstract*—This paper proposes Genetic Algorithm to find the shortest path for the graph mining problem. This algorithm uses the connection metrics of the given network to find the shortest path with the number of hops from the initial node to the destination node.**

Keywords—Genetic algorithm, chromosome, gene, algorithm, crossover, mutation

I. Introduction

Genetic Algorithm is a search heuristics, and is a process of selecting two randomly generated chromosomes to reproduce based on their fitness function. This reproduction will generate child chromosomes. The fitness function for every child chromosome is calculated to proceed with the formation of the future generations. The end result will be one of the optimal solution to find the shortest path from the initial node/start node to the destination node using this genetic algorithm.

II. Concept Design

A. Nodes/Vertex

The node is the individual points in the graph which is used to generate the graph.

B. Edges/Links

The line joining the two nodes are called as edges/links.

C. Gene

In this paper a gene is considered to be a node value.

D. Chromosome

A collection of node/genes form a chromosome. In this paper each chromosome is one of the paths connecting all the selected nodes.

E. Fitness

The fitness function for each chromosome is calculated based on the total number of hops between the initial node to destination/end node. The least number of hops will be the best fitness function.

F. Population

The chromosomes are randomly generated based on all possible path options connecting the required nodes. There will be 'n' available chromosomes in a population.

G. Reproduction

From the 'n' population only two chromosomes are selected as parent1(P1) and parent2 (P2) based on the fitness value. These two chromosomes are reproduced to generate the child chromosomes.

H. Crossover

The child chromosomes are formed by taking some parts of both the parents. The genes that are crossed over are selected in random for every generation.

I. Mutation

Mutation is process in which a probability of the original gene/ node would be replaced to generate a new child chromosome.

J. Population Replacement

The initial population is being replaced by the newly generated child chromosomes and their fitness function is calculated. The next set of parent nodes are selected from this newpopulation.

III. Discussion of Implementation

A. Overview of Implementation

The implementation of the graph mining problem using Genetic Algorithm is done in Python and makes use of the igraph python module to generate the graph with the nodes and links.

The graph is obtained by drawing the graph using the users input. The user should specify the start node, the in between node and the end node. Then based on the users input the initial graph highlights the user specified nodes.

Based on the nodes that the user specified, the genetic algorithm checks for all possible path options and

give one of the available solution which has the highest fitness function. The solution obtained is not the only optimal solution but it is one of the best optimal solutions available for the given nodes.

B. Heuristic

The Heuristics for estimating cost for this graph mining problem in genetic algorithm is the least number of hops. To get the best solution the minimum number of hops connecting the user specified nodes would be considered for the best fitness function.

C. Generic Algorithm

Genetic Algorithm is used to solve this graph mining problem. Firstly the initial population is created. This initial population consists of all the available path option for all the nodes that the user selected. Each individual path forms a chromosome and the nodes responsible for that particular path becomes the genes of that chromosome.

Secondly the fitness function is calculated for each chromosome. The best fitness function would be the chromosome with the lowest value. The fitness function is calculated based on the number of hops between each node to connect all the user specified nodes. The ideal solution for this graph mining problem is the path that has minimum number of hops connecting all the user specified nodes.

Based on the fitness function two chromosomes are chosen and considered as parent 1 (P1) and parent 2 (P2). Now the parents crossover or mutate to form their child chromosome. The child chromosome is added to the initial population.

Now again the genetic algorithm randomly picks two parents and crossovers or mutates resulting in reproduction of a new child chromosome. This process continues till 50 generations and tries to find one of the optimal solution for this graph mining problem.

D. *Algorithm*

The step by step process followed for the implementation of genetic algorithm is as follows[1]

- **[Start]** Generating the random population of 'n' chromosomes with all possible solution.
- **[Fitness]** Evaluate the fitness function *f(x)* of each chromosome *x* in the population. (For this paper f(x) is based on the number of hops it takes to connect all the specified nodes.)
- **[New population]** Create a new population by repeating following steps until the new population is complete
    - **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
    - **[Crossover]** The parents cross over to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
    - **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
    - **[Accepting]** Place new offspring in the new population
- **[Test]** If the end condition is satisfied, **stop**, and return the best solution in current population
- **[Loop]** Go to step **2**

E. *Implementation Diagram*

The basic implementation of this poker game is described in Fig.1.

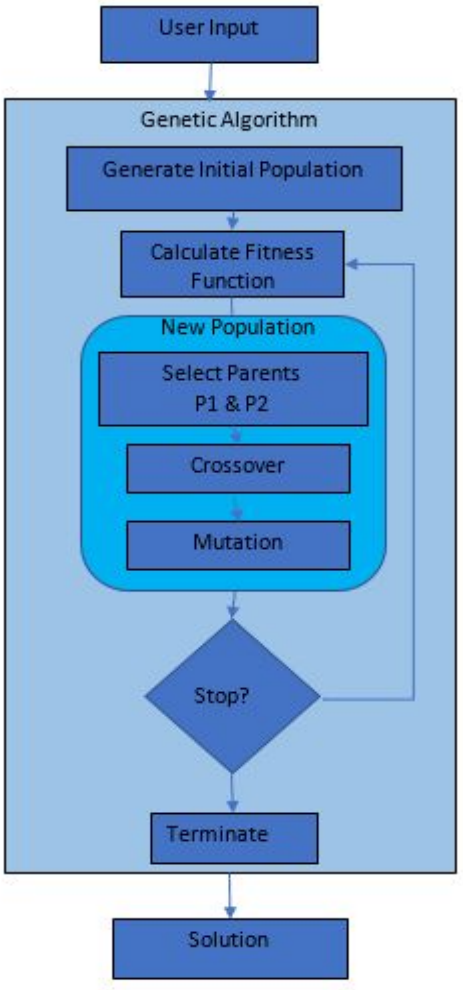

Fig.1. Process Flow Diagram for Graph Miner problem using Genetic Algorithm

F. Pseudocode

The pseudocode used to implement the graph mining problem using genetic algorithm is as follows

```
START
Generate the initial population
Compute fitness
REPEAT
        Selection
        Crossover
        Mutation
        Compute fitness
        UNTIL population has converged
STOP
```

Fig.2 Pseudocode for Genetic Algorithm[2]

G. Criterias for the Solution

The best solution for this graph mining problem is obtained by reproducing the selected parent chromosomes and crossovering them and mutating them to produce child chromosomes. Then these generated child chromosomes are verified for their fitness function. The fitness function includes the connections between all the nodes and the least number of hops. If one such solution is found then it is accepted to the newly generated population. The generated child node is also verified to the initial heuristic value of fitness to check if it is the optimal solution for the specified graph mining problem.

H. Running the Implemented Test

The user first specifies the nodes that are to be connected in this graph mining problem. Then the genetic algorithm checks for the initial population by checking for all the possible paths connecting all the user specified nodes. Individual sets of nodes that form a path to the solution form a chromosome. Thus the initial population is generated.

Then two chromosomes are picked from population with respect to the fitness function. These two chromosomes are the parents. Now the two parent chromosomes are cross overed to generate the child chromosomes. This crossover of the genes are done randomly. Now the generated child chromosome is mutated and is finally evaluated for the fitness function. Once the fitness function is calculated it is checked for the final criteria of the result, it gets added to the new population. If the solution has the best fitness function then that solution would be one of the optimal solutions for the graph mining problem.

There were two main test done to find if the implementation of the genetic algorithm was an optimal solution to get the best path between the user selected nodes.

The first experiment done to test the optimal solution was by limiting the number of generations. For this entire experiment the generations was varied from ten to fifty. The value of the generation was changed in the python code to check for the optimal solution. Because of this variation in the genetic algorithm, the parent chromosomes reproduces and generates child chromosomes only for the specified number of generation that is the process repeats only for the specified cycles.

The second experiment experiment was by limiting the population. The limit for the population to be generated varies each time of the execution of the genetic algorithm. By varying the population the genetic algorithm finds for the optimal solution in the set limit.

F. Test Results

Experiment-1

The result of the first experiment proved that the genetic algorithm used for the graph mining problem kept on giving different solutions when the generations was varied. The solution obtained would be one of the optimal solutions for the user specified paths. First the experiment was conducted in which the user gave four random nodes as when the implementation was limited to ten generations only The user gives input of nodes {4, 10,14,17}. Now the genetic algorithm is executed and the best solution is obtained . Fig. 3 shows how many chromosomes were generated in the population and it also show in how many hops the best solution was obtained. Finally the figure also show the path of one of the optimal solution for the graph mining problem.

```
jacobraj:Genetic jacobrajbenet$ python generic.py networkfile.txt 4 10 14 17

Number of arguments is 6
Number of nodes to traverse is 4

Initial Populations Size: 4695
First Path before applying Genetic Algorithm:
4 3 1 0 2 8 18 17 16 15 7 5 6 11 10 None
First distance before applying Genetic Algorithm: 15

Initial Populations Size: 3036
First Path before applying Genetic Algorithm:
4 3 1 11 10 None
First distance before applying Genetic Algorithm: 5

Initial Populations Size: 5955
First Path before applying Genetic Algorithm:
10 12 14 None
First distance before applying Genetic Algorithm: 3


Genetic Algorithm Finished...
Final Distance: 9
Final Solution:
[4, 3, 1, 11, 10, 12, 14, 13, 8, 17]
```

Fig.3 Population and Solution

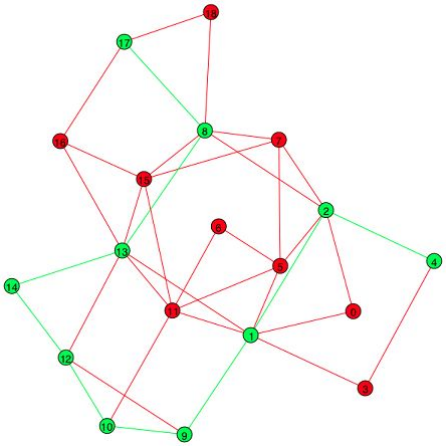

Fig.4 Final solution

Fig.4 shows the path highlighted in the initial graph using 10 generations. The green path is the best solution obtained by implementing genetic algorithm to the given graph with the selected nodes {4, 10,14,17}. Fig.3 and Fig.4 shows the optimal solution using ten generations.

Fig.5 and Fig.6 shows the optimal solution for connecting the same set of nodes {4, 10,14,17} using 50 generations.

```
jacobraj:Genetic jacobrajbenet$ python generic.py networkfile.txt 4 10 14 17

Number of arguments is 6
Number of nodes to traverse is 4

Initial Populations Size: 4695
First Path before applying Genetic Algorithm:
4 3 1 0 2 8 18 17 16 15 7 5 6 11 10 None
First distance before applying Genetic Algorithm: 15

Initial Populations Size: 3013
First Path before applying Genetic Algorithm:
4 3 1 11 10 None
First distance before applying Genetic Algorithm: 5

Initial Populations Size: 5955
First Path before applying Genetic Algorithm:
10 12 14 None
First distance before applying Genetic Algorithm: 3


Genetic Algorithm Finished...
Final Distance: 9
Final Solution:
[4, 2, 1, 9, 10, 12, 14, 13, 8, 17]
```

Fig.5 Population and Solution

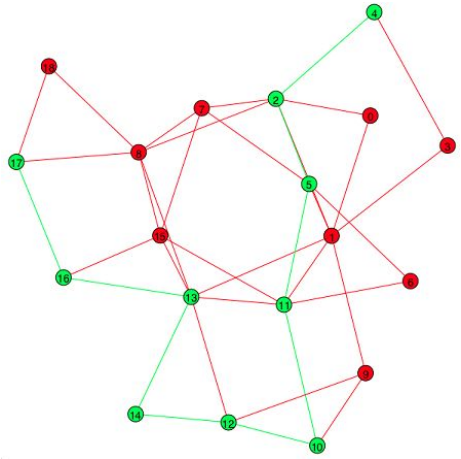

Fig.6 Final solution

Genetic algorithm is used to generate one of the optimal available solution, so as seen in Fig.4 and Fig.6 both are giving two different solutions for one graph mining problem of connecting the same set of nodes {4, 10,14,17}. The fitness function remains the same in both iterations.

Experiment-2

The result of the second experiment proved that the genetic algorithm used for the graph mining problem kept on giving different solutions when the population limit was varied. The solution obtained would be one of the optimal solutions for the user specified paths and would have different fitness function.. First the experiment was conducted in which the user gave four random nodes as when the implementation was limited to a population limit 1000. The user gives input of nodes {4, 10,14,17}. Now the genetic algorithm is executed and the best solution is obtained . Fig.7 shows how many chromosomes were generated in 1000 population limit and it also show in how many hops the best solution was obtained. Finally the figure also show the path of one of the optimal solution for the graph mining problem.

```
jacobraj:Genetic jacobrajbenet$ python generic.py networkfile.txt 4 10 14 17

Number of arguments is 6
Number of nodes to traverse is 4

Initial Populations Size: 1000
First Path before applying Genetic Algorithm:
4 3 1 0 2 8 18 17 16 15 7 5 6 11 10 None
First distance before applying Genetic Algorithm: 15

Initial Populations Size: 1000
First Path before applying Genetic Algorithm:
4 3 1 13 12 10 None
First distance before applying Genetic Algorithm: 6

Initial Populations Size: 1000
First Path before applying Genetic Algorithm:
10 11 1 13 14 None
First distance before applying Genetic Algorithm: 5


Genetic Algorithm Finished...
Final Distance: 16
Final Solution:
[4, 3, 1, 13, 12, 10, 11, 1, 13, 14, 12, 9, 10, 11, 13, 8, 17]
```

Fig.7 Population and Solution

Fig.8 shows the optimal solution obtained with a population limit of 1000. The solution seems to be big path.

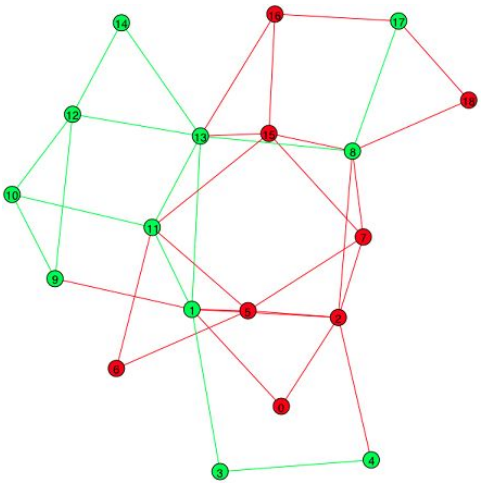

Fig.8 Final solution

Fig.9 and Fig.10 shows the optimal solution for connecting the same set of nodes {4, 10,14,17} using a population limit of 2000.

```
jacobraj:Genetic jacobrajbenet$ python generic.py networkfile.txt 4 10 14 17

Number of arguments is 6
Number of nodes to traverse is 4

Initial Populations Size: 2000
First Path before applying Genetic Algorithm:
4 3 1 0 2 8 18 17 16 15 7 5 6 11 10 None
First distance before applying Genetic Algorithm: 15

Initial Populations Size: 2000
First Path before applying Genetic Algorithm:
4 3 1 11 10 None
First distance before applying Genetic Algorithm: 5

Initial Populations Size: 2000
First Path before applying Genetic Algorithm:
10 9 12 14 None
First distance before applying Genetic Algorithm: 4


Genetic Algorithm Finished...
Final Distance: 13
Final Solution:
[4, 3, 1, 11, 10, 9, 12, 14, 12, 9, 1, 13, 16, 17]
```

Fig.9 Population and Solution

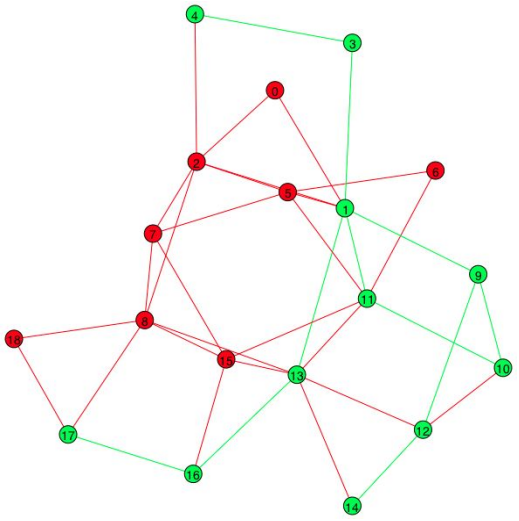

Fig.10 Final solution

Genetic algorithm is used to generate one of the optimal available solution, so as seen in Fig.8 and Fig106 both are giving two different solutions for one graph mining problem of connecting the same set of nodes {4, 10,14,17}. The more the population limit the less number of hops it takes for the nodes to give the optimal path. The fitness function differs in both the iterations.

IV. Conclusion

Genetic algorithm is used to find one of the optimal solutions available in the graph mining problem. The results of the first experiments conducted to prove shows two different solutions for the same set of nodes, but with the same fitness function. The result of the second experiment shows totally two different solutions with two different fitness functions. As a conclusion GEnetic Algorithm is used to find a solution for any problem, but the solution might not be the optimal solution. Limiting the population gives a worse solution with a higher fitness function.

V. References

[1] https://courses.cs.washington.edu/courses/cse473/06gorithmenetic Alhp/GeneticAlgDemo/gaintro.html

[2] https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

[3] https://medium.com/@becmjo/genetic-algorithms-and-the-travelling-salesman-problem-d10d1daf96a1

[4] http://www.theprojectspot.com/tutorial-post/applying-a-genetic-algorithm-to-the-travelling-salesman-problem/5