Power System Protection and Control
ISSN:1674-3415

AN EFFICIENT GENETIC ALGORITHM BASED METHOD FOR FINDING K SHORTEST AND LONGEST PATHS IN AN UNDIRECTED GRAPH

Siddhant Maske

Assistant Professor, Dept. of Computer Applications, D. Y. Patil school of Engineering and management,D.Y. Patil Education Society, (Deemed to be University)Kolhapur

Recevied: 22 December 2022 **Revised**: 25 January 2023 **Accepted**: 23 February 2023

ABSTRACT

This paper addresses genetic algorithm based method to find k shortest and longest path of a graph. The k-shortest path problem is a generalisation of the shortest path problem in graph theory. It has applications in areas like network routing and multi-object tracking. The longest path problem, on the other hand, is a generalisation of the Hamiltonian path problem. It is used in finding the critical path, among other things. This has applications in circuit design, planning and scheduling. The proposed genetic algorithm based method has been applied to both problems and the conditions which yield the maximum number of paths are determined. The proposed method is efficient to find out better quality path. Comparison with a traditional algorithm is made to judge the effectiveness of the approach .

Keywords*: Graph theory, shortest paths, longest paths, NP-hard, genetic algorithm, evolutionary programming, computer networks, scheduling algorithms*

INTRODUCTION

Graphs have application in every natural science. In computer science, graphs are used to represent computer networks, social media, and databases, etc. In physics and chemistry, graphs are used to study molecules. Graph theory is commonly used in molecular biology and genomics to model and analyse datasets with complex relationships. Graph theory has applications in other areas such as social sciences and linguistics.

A graph G is a pair (V, K). In a *simple undirected graph*, V is a set whose elements are called *vertices* (or *nodes*),and E is a set of two-sets of vertices, whose elements are called *edges*. The present study will be with simple, undirected, and weighted graphs with$|V| \geq 2$. Further, it will be labelled the elements of V using elements from the set of nonnegative integers,Z^{nonneg}. In a *weighted graph*, there is a mapping from E called the *weight*. We will use the set of positive integers,$\mathbb{Z}^{+}$, as the co-domain of this mapping. A *walk* is a sequence of edges which joins a sequence of vertices. A *path* is a walk in which all vertices are distinct. A *Hamiltonian path* is a path that visits each vertex exactly once. The *weight of a walk* in a weighted graph is the sum of the weights of the edges in the walk.

The *shortest path problem* is the problem of finding a path between two vertices (the *source* and the *destination*) such that the weight of the path is minimised. The *k shortest path problem* is the problem of finding additional short paths between the two vertices. To prevent unnecessary complexities, we will be labelling the nodes of our graph in such a way that the source node is always labelled 0.The k-shortest path problem has applications in network routing, multi-object tracking, power line engineering, etc. For example, in a network it is important to have alternate routing strategies when the shortest route is not feasible.

The *Hamiltonian path problem* is the problem of finding Hamiltonian paths in a graph. The *longest path problem* is a generalisation of this problem where we have a source node and a destination node, and we do not need to visit every vertex in the graph. It has applications in scheduling problems, among other things.Both the shortest path and k-shortest path problems are polynomial time. The Hamiltonian path problem on the other hand is NP-complete and the longest path problem is NP-hard [1].Evolutionary computation encompasses methods that simulate evolution. Algorithms that rely on evolutionary principles are called evolutionary algorithms. Evolutionary algorithms have applications in vast

DOI: 10.46121/pspc.51.1.2

areas, split into five broad categories namely planning, design, simulation and identification, control, classification in [2].*Genetic algorithms* (*GA*) are a type of evolutionary algorithm first described in[3].It is the emphasis placed on crossover which makes GAs distinctive[2]. We evolve a *population* of candidate solutions (*individuals*) towards better solutions. Each individual has a set of properties (*chromosomes*) which can be mutated and altered. In our case, every individual has exactly one chromosome.

LITERATURE REVIEW

Dijkstra's algorithm provides the most popular classical method of finding shortest paths [4],. The Bellman-Ford algorithm [5] is another algorithm for finding shortest paths. Both algorithms can be generalised to produce k-shortest paths. A. Schrijver [6] studies the history of shortest path algorithms further. When studying k-shortest path problem in particular, two variants are considered. In one variation, paths can visit the same node more than once. Using the terminology of this paper, I ought to call such "paths" walks with loops. In the other variation, paths are required to be simple and loop-less, these are actual paths according to our terminology. Classically, the loopy version is solved using

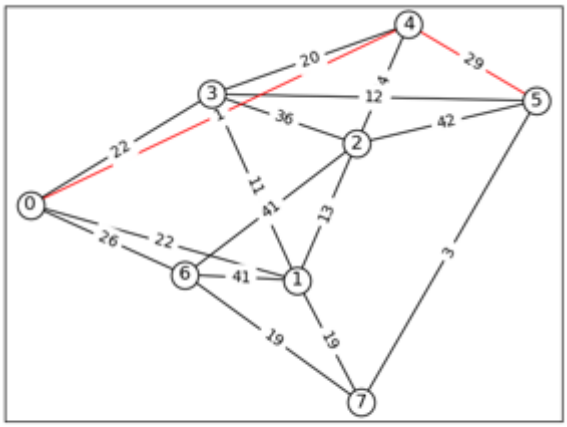


Figure 1 A path determined by the proposed algorithm

Eppstein's algorithm[7].The loop-less variation is solvable by Yen's algorithm[8]. The proposed algorithm in this paper produces a solution to the loop-less variation. Further discussion may be found in [9] and [10], among others. Evolutionary programming approaches for shortest path problems have been studied in[11], [12], [13], [14], among others. A. Y. Hamed [11] works on the loopy variation of the problem as described above. In F. Wang, Y. Man and L. Man [14] paper , the approach for encoding method and associated operators guarantees that the resulting population never has an individual representing an invalid path, or a path with loops. However, their method of generating the population and applying the mutation operator can be slow. I trade-off this performance hit with the fact that our population can have individuals representing invalid paths. These are discarded when considering their inclusion in the hall of fame. Dijkstra's[15] used longest path algorithm and goes into further details for efficient solutions. D. Karger, R. Motwani and G. D. Ramkumar [16] approximates longest paths. H. Bhasin and N. Gupta, [17] proposes a genetic algorithm to find the longest paths.

SOLUTION METHODOLOGY

The proposed methodology is genetic algorithm based to solve the objectives.

Encoding method

In a GA, the encoded chromosome is a solution to the problem in hand. In this algorithm, a chromosome is a string of vertex labels that excludes the source vertex label. In this description '0' is the source node of this string of vertex labels and find the weight of the walk to the destination node, if it exists.

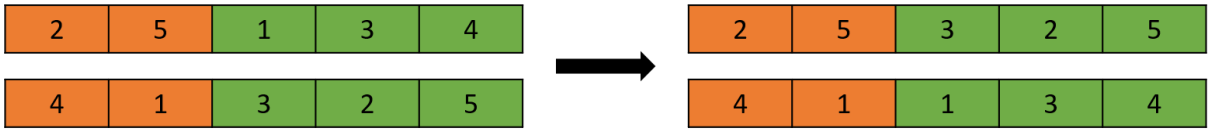
DOI: 10.46121/pspc.51.1.2

Initial population

A string containing the labels of every vertex other than the source vertex is randomly shuffled to generate the population.

Crossover

We pick a point in the given parents' chromosome (*Crossover point*).Nodes to the right of the point are swapped between the two individuals to produce the offspring. Nodes to the left of the point are kept as is.This is a conventional*one-point crossover*.



3.1.3 Mutation

The mutation operator selects a pair of genes from the chromosome to be mutated and swaps their positions in the string.

Selection

We use tournament selection, running tournaments (selecting the individual with the best fitness) among 3 randomly chosen elements from the previous population. Each spot in the new population is filled by running a tournament, the size of the population is kept constant across generations.

Evaluation for finding Shortest paths

The source node to the chromosome is selected and then determine the weight of the walk formed by the vertices in the chromosome until the first occurrence of the destination node. If the nodes do not form a walk, or if the destination node is not present in the chromosome, the fitness is set to infinity. Otherwise, the weight of the walk is the fitness. Individuals with infinite fitness are considered as *invalid*. The individuals are considered as *valid* otherwise. The objective is to minimise the fitness.

Evaluation for finding Longest paths

The source node to the chromosome is selected and then determine the weight of the walk formed by the vertices in the chromosome until the first occurrence of the destination node. If the nodes do not form a walk, or if the destination node is not present in the chromosome, the fitness is set to zero. Otherwise, the weight of the walk is the fitness.individuals with zero fitness is considered as *invalid*. The individuals are considered as *valid* otherwise. The objective is to maximise the fitness.

Algorithm

This algorithm satisfies all the requirements while being extremely easy to implement. A set of individual is added that contains the valid individuals seen throughout the generations.

Proposed Genetic Algorithm			
Input:	P_m P_c G P_{size}	← ← ← ←	Probability of mutation Probability of crossover Number of generations Population size
Output:	S	←	Set of paths
$t \leftarrow 0$ Initialise initial population $P(t)$			

DOI: 10.46121/pspc.51.1.2

Evaluation: Evaluate each individual in $P(t)$
Initialise the set$S$ with the valid individuals from $P(t)$
while$t \neq G$**do**
    $t \leftarrow t + 1$
    Selection: Generate $P(t)$ by runningP_{size} tournaments
    while we have enough individuals **do**
        Pick the first two individuals in G as I_1 and I_2
        if$random\ in\ [0,1) < P_c$**do**
            Crossover: Produce two new individuals, replacing I_1 and I_2 in G
        done
        Pick the next two individuals in G as I_1 and I_2
    done
    for$I$ in G**do**
        if$random\ in\ [0,1) < P_m$**do**
            Mutation: Produce a new individual, replacing I in G
        done
    done
    Evaluation: Evaluate each individual in $P(t)$
    Add all valid paths from $P(t)$ to S
Done

RESULT AND DISCUSSION

Maximising yield

After applying the proposed method on a Erdős–Rényi graph [18] , the result has been prepared. Implementation of all the algorithms was done in PYTHON. The method has been applied on the graph consisting of 90 nodes. All the programs have been executed on core-i3 (8th generation) machine with 6GB RAM and results have been reported accordingly. The probability of an edge existing between two nodes is set to 0.5, and the edges may have any weight from 1 through 50 inclusive. A population size is set to 500 and number of generations is set to 20,000 . Generation wise, number of unique paths obtained by varying mutation probability P_m from 0.2 to 0.8 where cross over probability P_c was kept fixed at 0.3. The result has been shown graphically in Figure 2.

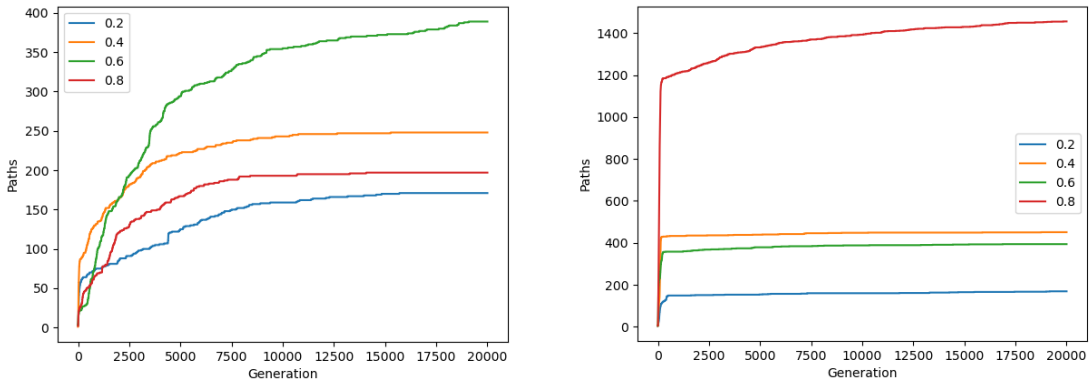


Figure 1: Paths-Generation graph with P_m as parameter (left – minimisation, right – maximisation)

The behaviour was found to be consistent across different runs and higher values of P_m corresponded to higher number of paths generated.

DOI: 10.46121/pspc.51.1.2

Again , number of unique paths obtained by varying crossover probability P_c from 0.1 to 0.9 where cross over probability P_m was kept fixed at 0.8. The result has been shown graphically in figure 3.

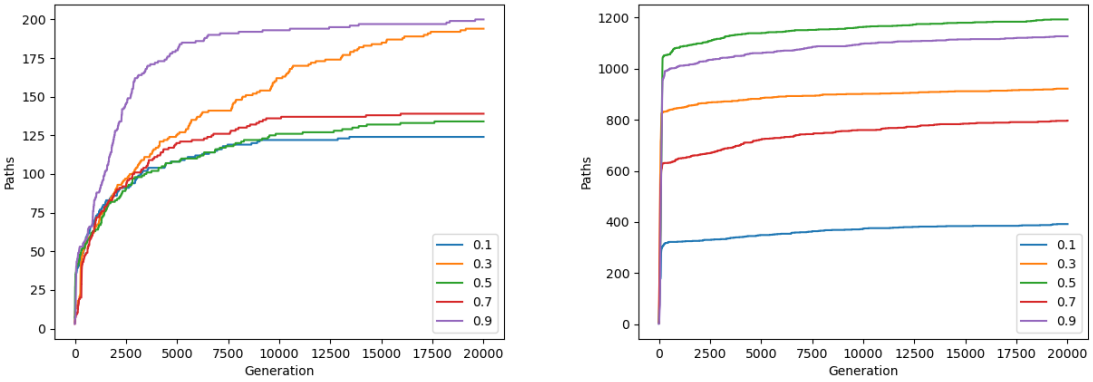


Figure 2: Paths-Generation graph with P_c as parameter (left – minimisation, right – maximisation)

The behaviour appeared significantly inconsistent across different runs. No value of P_c was found to be better than the rest.

Comparison of the shortest path variation against Yen's algorithm

The quality of paths produced by the proposed algorithm with the traditional Yen's algorithm used to determine k-shortest paths Figure 3. Here,P_m was kept fixed at 0.8 and $P_c$at 0.3.

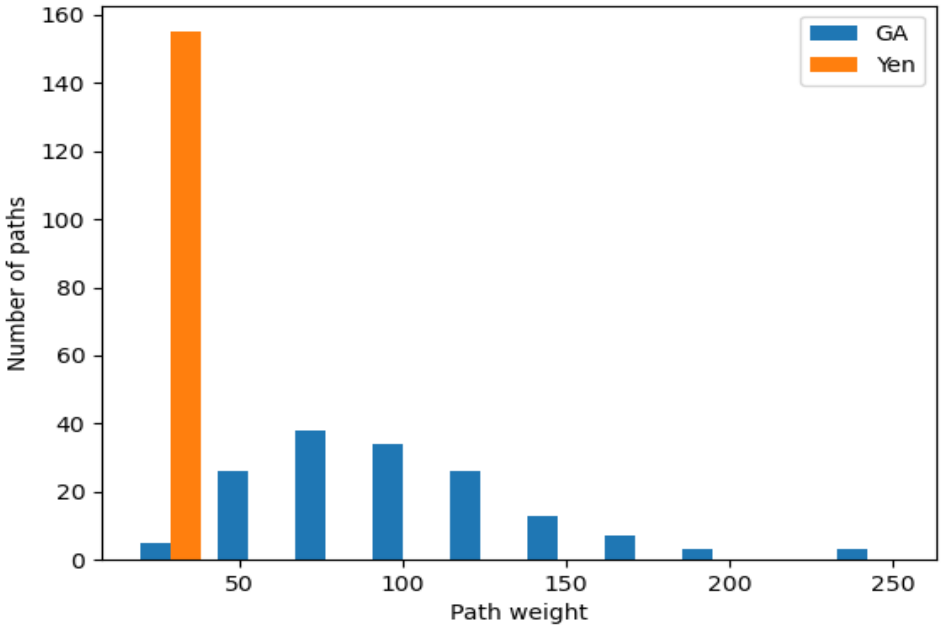


Figure 3: Histogram of weights of paths obtained

DOI: 10.46121/pspc.51.1.2

The proposed method took higher time than Yen's algorithm to produce the same number of paths. The quality of the paths produced is significantly better than the classical algorithm.

CONCLUSION

The paper proposed a genetic algorithm to determine the k shortest and longest paths in a simple undirected graph. The algorithm tries to minimise the resources needed to generate such paths. It does this by structuring the problem as a general GA problem. The method is based on the principle of Darwin's theory. The main philosophy is followed to Holland . The probability mutation has been varied and variation of cross over probability has been done to achieve good quality solution . Generation wise the paths have been observed in every case. The quality of the paths produced by this method is significantly better than the classical algorithm The solution to the longest path problem is of more interest because of the complexity of the problem. Worst case complexity of the proposed method is calculated by looking at the nature of the loops. And finally the complexity of the /algorithm is $O(G \times (P_{size} + P_c \times (P_{size} \div 2) + P_m \times P_{size})$. The proposed method is slower than Yen's algorithm, so this method will be modified in future to reduce runtime of the algorithm.

REFERENCES

1. N. Deo, Graph theory with applications to engineering and computer science, Courier Dover Publications, 2017.
2. T. Bäck, D. B. Fogel and Z. Michalewicz, Evolutionary computation 1: Basic algorithms and operators, CRC press, 2018.
3. J. H. Holland, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, MIT press, 1992.
4. E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik,* vol. 1, no. 1, pp. 269-271, 1959.
5. R. Bellman, "On a routing problem," *Quarterly of applied mathematics,* vol. 16, no. 1, pp. 87-90, 1958.
6. A. Schrijver, "On the history of the shortest path problem," *Documenta Mathematica,* vol. 17, no. 1, pp. 155-167, 2012.
7. D. Eppstein, "Finding the k shortest paths," *SIAM Journal on computing,* vol. 28, no. 2, pp. 652-673, 1998.
8. J. Y. Yen, "Finding the k shortest loopless paths in a network," *management Science,* vol. 17, no. 11, pp. 712-716, 1971.
9. B. L. Fox, "κth shortest paths and applications to the probabilistic networks," *ORSA/TIMS Joint National Mtg.,* vol. 23, p. B263, 1975.
10. J. Hershberger, M. Maxel and S. Suri, "Finding the k shortest simple paths: A new algorithm and its implementation," *ACM Transactions on Algorithms (TALG),* vol. 3, no. 4, pp. 45-es, 2007.
11. A. Y. Hamed, "A genetic algorithm for finding the k shortest paths in a network," *Egyptian Informatics Journal,* vol. 11, no. 2, pp. 75-79, 2010.
12. A. A. Heidari and M. R. Delavar, "A MODIFIED GENETIC ALGORITHM FOR FINDING FUZZY SHORTEST PATHS IN UNCERTAIN NETWORKS.," *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences,* vol. 41, 2016.

DOI: 10.46121/pspc.51.1.2

13. L. Liu, H. Mu, X. Yang, R. He and Y. Li, “An oriented spanning tree based genetic algorithm for multi-criteria shortest path problems,” *Applied soft computing,* vol. 12, no. 1, pp. 506-515, 2012.
14. F. Wang, Y. Man and L. Man, “Intelligent optimization approach for the k shortest paths problem based on genetic algorithm,” in *2014 10th International Conference on Natural Computation (ICNC)*, 2014.
15. R. Uehara and Y. Uno, “Efficient algorithms for the longest path problem.,” in *International symposium on algorithms and computation*, Berlin, Heidelberg, 2004.
16. D. Karger, R. Motwani and G. D. Ramkumar, “On approximating the longest path in a graph,” *Algorithmica,* vol. 18, no. 1, pp. 82-98, 1997.
17. H. Bhasin and N. Gupta, “Critical path problem for scheduling using genetic algorithm.,” in *Soft Computing: Theories and Applications*, Singapore, 2018.
18. P. Erdős and A. Rényi, “On Random Graphs I,” *Publicationes Mathematicae,* vol. 6, pp. 290-297, 1959.

DOI: 10.46121/pspc.51.1.2