A Fast and Effective Technique Based on A Genetic Algorithm for Determining the K Shortest and Longest Path

Sharadindu Roy
[1]Department of Computer Science, Sonarpur Mahavidyalaya,
Affiliated to University of Calcutta, Kolkata-700149, West Bengal, India

***Abstract*- This paper discusses a genetic algorithm-based method for determining the k shortest and longest paths in a graph. The k-shortest path problem is a graph theory generalisation of the shortest path problem. It has applications in network routing and multi-object tracking, among others. In contrast, the longest path problem is a generalisation of the Hamiltonian path problem. It is used, among other things, to find the critical path. This can be used in circuit design, planning, and scheduling. The proposed genetic algorithm-based method was applied to both problems, and the conditions that yielded the most paths were determined. The proposed method is effective for determining a better-quality path. To assess the effectiveness of the approach, it is compared to a traditional algorithm.**

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

1. Introduction

Graphs are used in every science. Graphs are used in computer science to represent computer networks, social media, and databases, among other things. Graphs are used to study molecules in physics and chemistry. In molecular biology and genomics, graph theory is commonly used to model and analyse datasets with complex relationships. A graph G is a pair of graphs (V, K). In a simple undirected graph, V is a set of vertices (or nodes), and E is a set of two-sets of vertices, the elements of which are called edges. The current investigation will use simple, undirected, and weighted graphs with $|V|\geq 2$. It will also label the elements of V with elements from the set of nonnegative integers, Z^{nonneg}. The weight is the mapping from E in a weighted graph. As the co-domain of this mapping, we will use the set of positive integers, Z^{+}. A walk is a set of edges that connects a set of vertices. A path is a walk that has distinct vertices. A Hamiltonian path is one that goes through each vertex exactly once. In a weighted graph, the weight of a walk is the sum of the weights of the edges in the walk. The shortest path issue is the challenge of finding the shortest path between two vertices (the source and the destination) while keeping the path's weight to a minimum. The problem of finding additional short paths between two vertices is known as the k shortest path problem. To avoid unnecessary complications, we will label the nodes of our graph so that the source node is always labelled 0. The k-shortest path problem is used in network routing, multi-object tracking, power line engineering, and other fields. In a network, for example, it is critical to have alternate routing strategies in place when the shortest route is not feasible.

The problem of finding Hamiltonian paths in a graph is known as the Hamiltonian path problem. The longest path problem is a variant of this problem in which we have a source and a destination node and do not need to visit every vertex in the graph. It can be used to solve scheduling problems, among other things. The shortest path and k-shortest path problems are both polynomial time problems. The Hamiltonian path problem is NP-complete, whereas the longest path problem is NP-hard [1]. Methods for simulating evolution are included in evolutionary computation. Evolutionary algorithms are algorithms that are based on evolutionary principles. Evolutionary algorithms have a wide range of applications, which can be divided into five broad categories: planning, design, simulation and identification, control, and classification [2]. Genetic algorithms (GA) are a class of evolutionary algorithms that were first described in[3]. GAs are distinguished by the emphasis placed on crossover[2]. We improve solutions by evolving a population of candidate solutions (individuals). Each person has a unique set of properties (chromosomes) that can be mutated and altered. In our case, each person has exactly one chromosome.

2. Related works

The most popular classical method of finding shortest paths is Dijkstra's algorithm [4]. Another algorithm for finding shortest paths is the Bellman-Ford algorithm [5]. Both algorithms can be extended to find the k-shortest path. A. Schrijver [6] delves deeper into the history of shortest path algorithms. Two variants are considered when studying the k-shortest path problem in particular. Paths can visit the same node multiple times in one variant. Using the terminology of this paper, such "paths" should be referred to as walks with loops. Paths in the other variation must be simple and loop-free; these are actual paths in our terminology. Eppstein's algorithm[7] is traditionally used to solve the loopy version. Yen's algorithm[8] can solve the loop-less variation. The proposed algorithm in this paper solves the loop-less variation. Further discussion can be found, among other places, in [9] and [10]. [11], [12], [13], and [14] have all investigated evolutionary programming approaches for shortest path problems. A. Y. Hamed [11] is working on the above-mentioned loopy variation of the problem. The approach for encoding method and associated operators proposed by F. Wang, Y. Man, and L. Man [14] ensures that the resulting population never contains an individual representing an invalid path or a path with loops. Their method of generating the population and applying the mutation operator, on the other hand, can be slow. I balance this performance hit with the possibility of individuals representing invalid paths in our population. When considering their inclusion in the hall of fame, these are discarded. Dijkstra's[15] longest path algorithm is used, and he goes into

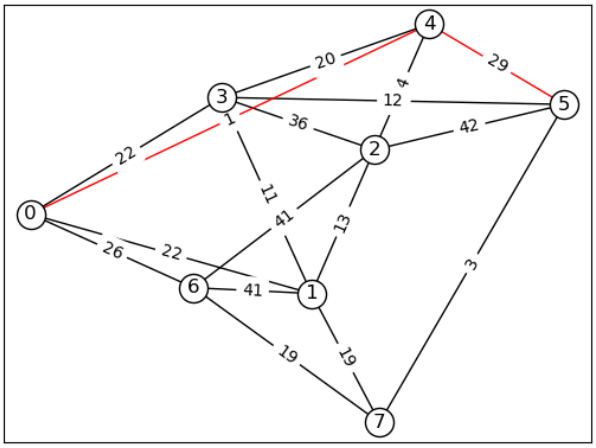


Figure 1 A path determined by the proposed algorithm

more detail for efficient solutions. Longest paths are approximated by D. Karger, R. Motwani, and G. D. Ramkumar [16]. [17] H. Bhasin and N. Gupta propose a genetic algorithm for determining the longest paths.

3. Solution methodology

To achieve the objectives, the proposed methodology uses a genetic algorithm.

3.1 Encoding method

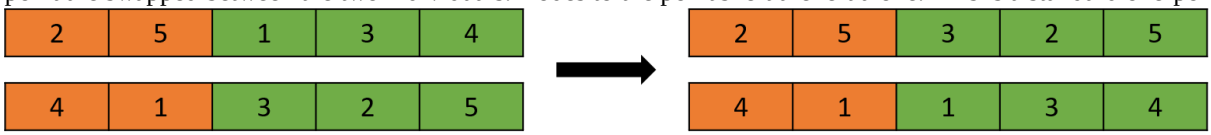
The encoded chromosome is a solution to the problem at hand in a GA. A chromosome is defined in this algorithm as a string of vertex labels that excludes the source vertex label. '0' is the source node of this string of vertex labels in this description; find the weight of the walk to the destination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

3.1.1 Initial population

To generate the population, a string containing the labels of all vertex except the source vertex is randomly shuffled.

3.1.2 Crossover

selecting a location on the chromosomes of the given parents (Crossover point). To produce the offspring, nodes to the right of the point are swapped between the two individuals. Nodes to the point's left are left alone. This is a standard one-point crossover.



3.1.3 Mutation

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

3.1.4 Selection

We employ tournament selection, which entails running tournaments (selecting the individual with the best fitness) among three randomly selected elements from the previous population. Each new population spot is filled by holding a tournament, and the population size remains constant across generations.

3.1.5 Evaluation for determining the shortest paths

The source node to the chromosome is chosen, and the weight of the walk formed by the vertices in the chromosome until the first occurrence of the destination node is determined. The fitness is set to infinity if the nodes do not form a walk or if the destination node is not present on the chromosome. Otherwise, the fitness is the weight of the walk. Individuals with infinite fitness are deemed ineligible. Otherwise, the individuals are considered valid. The goal is to reduce fitness as much as possible.

3.1.6 Evaluation for determining the longest paths

The source node to the chromosome is chosen, and the weight of the walk formed by the vertices in the chromosome until the first occurrence of the destination node is determined. The fitness is set to zero if the nodes do not form a walk or if the destination node is not present on the chromosome. Otherwise, the fitness is the weight of the walk. Individuals with no fitness are deemed invalid. Otherwise, the individuals are considered valid. The goal is to achieve maximum fitness.

3.2 Algorithm

This algorithm meets all of the requirements while being extremely simple to implement. A set of individuals is added that includes the valid individuals seen across generations.

Proposed Genetic Algorithm	
Input:	P_m ← Probability of mutation P_c ← Probability of crossover G ← Number of generations P_{size} ← Population size
Output:	S ← Set of paths

$t \leftarrow 0$
Initialise initial population $P(t)$
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

4. RESULT AND DISCUSSION:

4.1 Maximising yield

The result has been prepared after applying the proposed method to an Erds-Rényi graph [18]. PYTHON was used to implement all of the algorithms. The method has been applied to a graph with 90 nodes. All of the programmes were run on a core-i7 (10th generation) machine with 8GB RAM, and the results were reported accordingly. The probability of an edge forming between two nodes is set to 0.5, and edges can have any weight between 1 and 50. The population is limited to 500 people, and the number of generations is limited to 20,000. Number of unique paths obtained by varying mutation probability P_m from 0.2 to 0.8 while keeping cross over probability P_c constant at 0.3. Figure 2 depicts the outcome graphically.

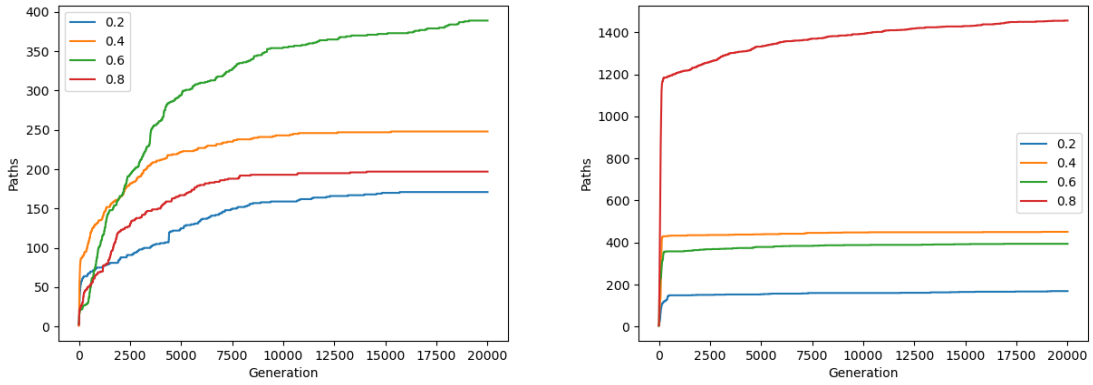


Figure 2: Paths-Generation graph with P_m as parameter (left – minimisation, right – maximisation)

The behaviour was found to be consistent across runs, and higher P_m values corresponded to a greater number of paths generated. Again, the number of distinct paths was obtained by varying the crossover probability P_c from 0.1 to 0.9 while keeping the cross over probability P_m constant at 0.8. Figure 3 depicts the outcome graphically.

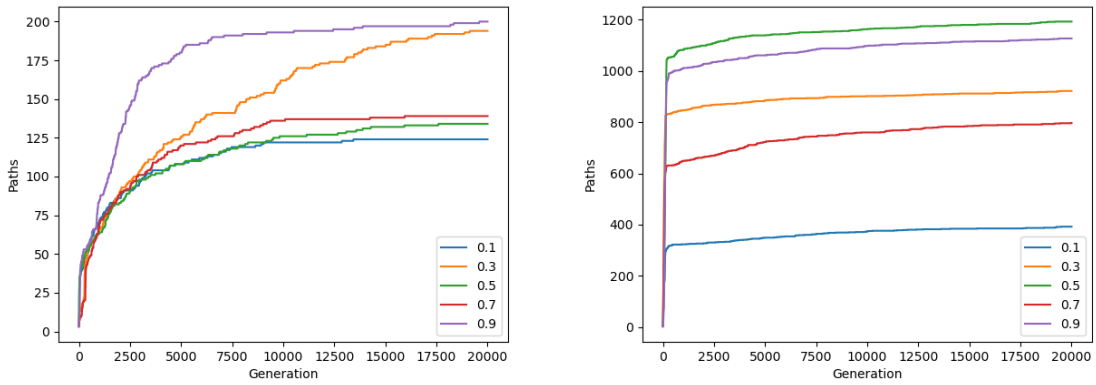


Figure 3: Paths-Generation graph with P_c as parameter (left – minimisation, right – maximisation)

The behaviour appeared to be significantly inconsistent across runs. No P c value was discovered to be superior to the others.

4.2 Evaluation of the shortest path variation versus Yen's algorithm

Figure 4 compares the quality of paths produced by the proposed algorithm to the traditional Yen's algorithm used to determine the k-shortest paths. P_m was set to 0.8 and Pc at to 0.3 in this case.

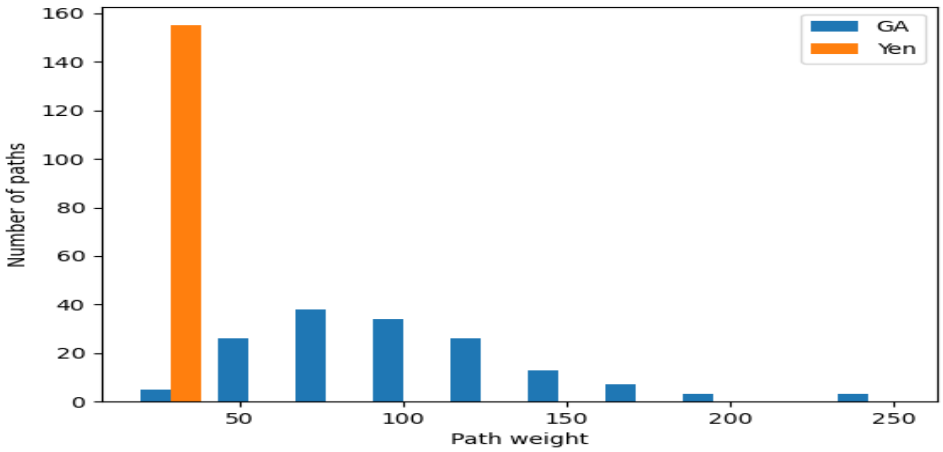


Figure 4: Histogram of weights of paths obtained

To generate the same number of paths, the proposed method took longer than Yen's algorithm. The quality of the paths produced is noticeably higher than that of the classical algorithm.

5. CONCLUSION:

A genetic algorithm was proposed in the paper to find the k shortest and longest paths in a simple undirected graph. The algorithm attempts to reduce the resources required to generate such paths. This is accomplished by structuring the problem as a general GA problem. The method is based on Darwin's theory of evolution. Holland adheres to the main philosophy. To achieve a good quality solution, the probability mutation has been varied, as has the cross over probability. The paths have been observed generationally in every case. The quality of the paths produced by this method is noticeably higher than that of the classical algorithm. Because of the problem's complexity, the solution to the longest path problem is more interesting. 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 result clearly demonstrates the efficacy of the proposed method. In terms of solution quality, the proposed method outperforms Yen's 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.
[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.

Sharadindu Roy is currently employed as an Assistant Professor and Head at Sonarpur Mahavidyalaya in Raipur, South 24 Parganas, Kolkata, and West Bengal, India. In 2005, he received a postgraduate degree in computer science from the University of Calcutta. He has numerous papers published in national and international journals. His research focuses on the mechanism of VLSI physical design and the development of algorithms to solve the problem of integrated circuit (IC) partitioning. He is also fascinated by soft computing and optimisation. He wishes to investigate various multi-objective problems and apply a soft computing approach to them.