IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

A Study Of Networks Route Optimization Through Graph Algorithms

[1]Tarni Sahu, [2]Renuka Sahu

[1]Research Scholar, [2]Assistant Professor

[1,2]Department of Mathematics

[1,2] Kalinga University, Naya Raipur, Chhattisgarh, India.

Abstract: Nowadays, in computer networks, the routing is based on the shortest path problem. This will help in minimizing the overall costs of setting up computer networks. New technologies such as map-related systems are also applying the shortest path problem. Optimization network routes using graph theory involves analyzing network structures as graphs and applying algorithms to find the most efficient paths or configuration. In this work we have study and analysis the Djikstras' Algorithm, Floyd-Warshall's Algorithm, Bellman-Ford Algorithm, Breadth-First Search (BFS), Johnsons Algorithms. These algorithms are used in different routing techniques in networking like Mobile ad-hoc, V-ad-hoc, sensor network etc. This work also aims to promote additional investigate on topics of route optimization and concludes with several suggestions for further research.

Index Terms - Djikstras' Algorithm, Floyd-Warshall's Algorithm, Bellman-Ford Algorithm, Breadth-First Search (BFS), Johnsons Algorithms**.**

I. INTRODUCTION

A diagram, which is composed of a set of points along with lines connecting specific pairings of these points, can be used to represent a wide range of real-world scenarios or issues. This real-world scenario's mathematical abstraction results in the notion of a graph. A graph is a type of mathematical construction where several points connect in order to represent a certain function or rule. The lines that are linking the points may be referred to as sides or edges, while the points themselves may be referred to as vertices. The study of graph theory enables us to understand how to construct graphs and solve problems with them. The investigation of graphs, which are mathematical patterns that show pair relationships between objects, is known as graph theory. In this dissertation we study basics of graph theory. As well as its application in computer network to find optimized route.

The word "network" is typically used to describe situations in which information needs to be sent or transported between nodes via links, be they information in any network or physical objects connected by roads, trains, or other means. A network can be defined as a group of interconnected items that facilitate the flow of various commodities, including people, products, information etc. In reality, a data communication networks can represents a wide range of systems which are connected and share the information. Several instances include the internet, the World Wide Web, social media, citation-linked publication networks, transportation, electrical and communication networks etc. A diagram that results from replacing every node in a network with lines that have dots or circles at both ends is called a graph.

Communication networks can be expressed using graph theory. A collection of terminals, links, and nodes that interact to enable communication amongst terminal clients makes up a communication network. To illustrate communication networks, graphs are essential . In most cases, the vertices of a graph represent terminals, processors, and the edges, cables, filaments, and other transmission routes that the data passes through. Below is a quick review of graphs and communication networks; Next chapter

provides a more thorough discussion of graph theory in communication networks. Graphs are useful for modeling communication networks because they may be used to depict communication networks.

A graph theory problem where vertices represent locations (like stops or schools) and edges represent the connections between them (like roads or paths). GPS and Google Maps indeed use algorithms based on graph theory to find the shortest path between two points efficiently. In context, a Hamiltonian path is a path in a graph that visits each vertex exactly once. However, finding a Hamiltonian path in a general graph is an NP-complete problem, which means it's computationally very hard to solve efficiently for large graphs. In the context of schools and colleges using such systems to transport students, the software optimizes routes based on various factors such as traffic conditions, distance, and time constraints. The goal is to efficiently transport students from their stops to schools while minimizing travel time and maximizing resource utilization.

Graph theory is applicable to many different types of networks like computer network. The purpose of the present study is to examine the significance and applicability of principles of graph theory to analysis for finding optimum route on communication networks. The present work explains how graph theory is applied in different computer networks routing, which aids in the effective representation of graph theory. Our main objective of this work is to study and analysis of various graph algorithms for finding shorted path in terms of minimum time and minimum cost.

II. EVOLUTION OF GRAPH THEORY

The famous Swiss mathematician Leonhard Euler is acknowledged with discovering graph theory. Euler proposed to deal with the Konigsberg Bridge problem in 1735. This problem turned out to be a riddle that worried about the possibility of solving it by using each and every one of the seven bridges that were constructed over a river that flows through a small island. It was required that a scaffold not be passed twice. Euler proved that such a method does not exist. His verification represented an overview of graph theory [1].

This is regarded as the subject's genesis. Similarly, William R. Hamilton, the investigator, also thought about polyhedral cycles, which led to the original inspiration for two of Hamiltonian theory: it is necessary to identify a path around a graph that passes through each vertex. The era of Hamiltonian cycles began with this. It was approximately a century for the primary source material on graph theory [2].

Cayley was motivated to conduct research about specific types of graphs, called trees, by his affinity with certain analytical forms that originated through differential analytics. The techniques he employed mainly deal with the identification of graphs with certain features. At that point, the outcome of Cayley and the significant findings dispersed by Polya between 1935 and 1937 brought shape to enumerative graph theory [3]. Cayley made a connection between what he discovered on trees and the latest studies on synthetic components. The combination of mathematical and scientific ideas continued to take form and eventually discovered ways to integrate into the standard language of graph theory.

More specifically, Sylvester introduced the term "graph" in 1878 [4]. Denes Konig created and released the fundamental source material on graph theory in 1936 [5]. Frank Harary published another book in 1969 that allowed social scientists, mathematicians, scientists, and electrical architects to communicate with one another [6]. However, several additional instances occurred that raised interest in graph theory and led to new studies in the field.

EXPLORATION OF GRAPH THEORY: TERMINOLOGIES OF GRAPH FOR DIFFERENT NETWORKS

Def. 1.a. A graph G is an ordered pair (V(G), E(G)) consisting of a set V(G) of vertices and a set E(G), disjoint from V(G), of edges, mutually with an occurrence function F(G) that associates with each edge of G an unordered pair of vertices of G. If E is an edge and u and v are vertices such that F(G(e)) = {u,v}, then E is said to connect u and v, and the vertices u and v are called the ends of e. Here denote the numbers of vertices and edges in graph by V(G) and E(G); It is two basic parameters i.e. the order and size of graph. In a simple graph there are no parallel edges and no selfl-loop.

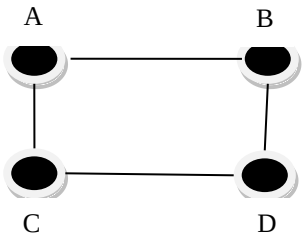

Fig.1.: Simple Graph

Def. 1.b. The number of edges occurring on a vertex v is its degree. The number of edges that incident on a vertex vi in graph G is its degree, which is represented by the symbols di, d(v), or degvi. The symbols δ(G) and Δ(G) represent the minimum and maximum degrees of a vertex in G. The "degree" of a vertex in a graph refers to the number of edges incident to that vertex. In other words, it's the count of how many edges are connected to a particular vertex.

There are two types of degrees commonly discussed in graph theory:

Vertex Degree: This refers to the number of edges incident to a vertex. In a directed graph, we have an in-degree (number of edges coming into the vertex) and an out-degree (number of edges going out from the vertex).
Graph Degree: This refers to the maximum degree of any vertex in the graph. It gives us a sense of the connectivity of the graph.

In some contexts, "degree" may refer to the number of vertices connected to a particular vertex, which is different from the number of edges. Understanding the degrees of vertices and graphs is fundamental in graph theory, as it helps in analyzing the structure and properties of graphs.

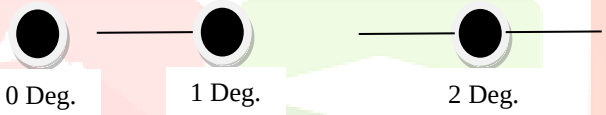

Fig.2.: Graph with different degrees

Def. 1.3. A collection of vertices V = {v1, v2, …} and a set of edges E = { e1, e2, …} and a mapping function that maps each edge onto an ordered pair of vertices (v_i, v_j) make up a directed graph, digraph, or oriented graph G.

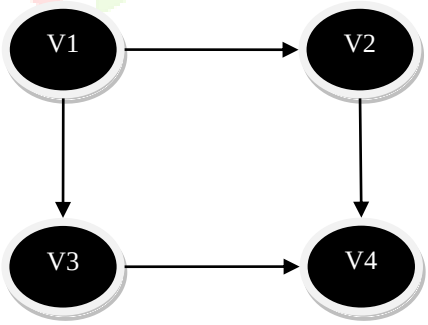

Fig.3.: Directed Graph

Def. 1.4.A graph G is considered complete if each vertex is next to every other vertex in G.
K_n represents a complete graph with n vertices. That is to say, full graphs which have every hub ('u') in graph 'G' neighboring every other hub ('v') are frequently referred to as universal graphs. With “n (n-1)/2” edges, a graph is complete.
Def. 1.5 A graph in cycles Cn is a connected graph with n vertices in which degree 2 is assigned to each vertex. There is just one cycle in this kind of graph.
Def. 1.6 If a graph G has a vertex set that can be divided into two sets, X and Y, with one end vertex in X and the other in Y, then the graph is bipartite. Here, X and Y are referred to as the partite sets.

Def.1.7 In a linked network, a Hamiltonian path is a path that contains each graph vertex precisely once. A circuit is referred to as a Hamiltonian circuit if every graph vertex—aside from the start and last vertices—is contained exactly once. If a graph has a Hamiltonian circuit or path, it is called a Hamiltonian graph.

Def. 1.8 K-dimensional cube or hypercube The simple graph Qk is made up of pairs of k-tuples that differ by exactly one place, and its vertices are k-tuples with entries in {0,1}.

Def. 1.9 When a graph G is graphed, a walk is given by the sequence W: = v_0, e_1, v_1...v_{n-1} e lvl, where terms are vertices and edges of G in turn (not necessarily distinct), and v_{i-1} and v_i are the ends of e_i for $1 < i \leq l$. In a graph, a walk is considered closed if its initial and terminal vertices are the same, and it is a trail if each of its edge terms is unique. An Euler trail is a path that travels down every edge of a graph.

Def 1.10 One of the most famous examples of a non-planar graph is the complete graph K5$K5$ (the graph with five vertices where each vertex is connected to every other vertex by an edge) and the complete bipartite graph $K_{3,3}$ (the graph with two sets of three vertices, where each vertex in one set is connected to every vertex in the other set by an edge).The Kuratowski theorem states that a graph is non-planar if and only if it contains a subgraph that is homeomorphic to K_5 or $K_{3,3}$. This theorem provides a convenient criterion for determining whether a graph is planar or non-planar. Non-planar graphs have various applications in graph theory, computer science, and other fields, and they are studied for their properties and limitations in different contexts.

Def. 1.11 The graph with vertices set V in which two vertices (x_1,y_1) and (x_2,y_2) are near if their Euclidean distance equals 1 is the unit distance graph on a subset V of R^2. Let f be a face in a planar embedding of G, and let G be a planar graph. Next, a planar embedding with an outer face that shares the same boundary with f is admitted by G.

Any plane graph's dual is connected. The Euler's formula for a connected plane graph G is

$$v(G) - e(G) + f(G) = 2.$$

The meaning of a line graph The simple graph L(G) has ef E(L(G)) when e and f share a common endpoint in G. The vertices of L(G) are the edges of G. The intersection graph of (V, F) is referred to as an interval graph when V = R and F is a collection of closed intervals of R. A family of intervals allocated to the vertices of a graph, such that vertices are adjacent if and only if the associated intervals intersect, is the interval representation of the graph. A graph using this kind of representation is called an interval graph.

Def. 1.12 An open walk in which no vertex appears more than once is called path. The number of edges in the path is called length of a path, a path from vertex A to vertex M is shown below. It is one of many possible paths in this graph.

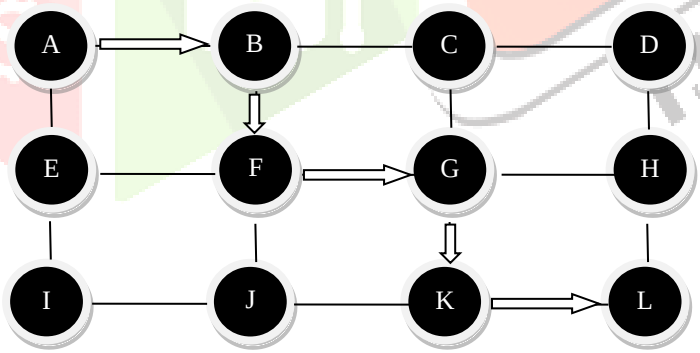

Fig: Path in graph

Def. 1.13 If a path connects each pair of vertices in a graph, then it is said to be linked. If a path exists from any vertex to another vertex, then the two are associated with a graph.

Def.1.14 A cycle is a graph made up of only one circuit. An odd number of vertices in a cycle is referred to as an odd cycle, and an equal number of vertices in a cycle is called an even cycle.

Def.1.15 A cycle is a graph made up of only one circuit. An odd number of vertices in a cycle is referred to as an odd cycle, and an equal number of vertices in a cycle is called an even cycle.

Def.1.16 All of the links in the cycle go in the same general direction. Within a transportation system, circuits are crucial.

Def.1.17 A graph that has numerical weights assigned to each branch is called a weighted graph. Thus, a weighted graph is a particular kind of labelled graph where the labels are numerical. The distance between two sites, the expense of the journey, or the time it takes could all be represented by the weights. It is crucial that an edge's weights and the distance between a graph's vertices be not exactly comparable.

The above definitions are for understanding the graph theory and their different terminology.

NETWORK ANALYSIS THROUGH GRAPH THEORY

The term "network" is typically used to describe situations in which information needs to be sent or transported between nodes via links, be they information in any network or physical objects connected by roads, trains, or other means. A network is just a group of interconnected items that facilitate the movement or flow of various commodities, including people, products, information, and current. In the actual world, a network can represent a variety of systems. The internet, the World Wide Web, social networks, networks of publications connected by citations, networks of transportation, metabolic, electrical, and communication networks etc. are a few examples. A graph is created when all of the components in a network are changed to lines with dots or circles at both ends.

III. LITERATURE REVIEW

There are number of algorithms are proposed and analyzed. Here we presented the review of research in field of computer network route optimization.

Natarajan Meghanathan et. al [7] discussed the Bellman-Ford and Dijkstra algorithms for determining the shortest path in a graph. Finally, he says that the temporal complexity for the time complexity of the Bellman-Ford algorithm is O(|V||E|), while that of the Dijkstra algorithm is O(|E|*log|V|). As per the findings of *Lili Cao et. al [8]*, finding the shortest pathways is a fundamental primitive for various graph-based applications, especially those on virtual social networks. For instance, in order to facilitate introductions, LinkedIn users do searches to determine the "social link" that will connect them to a specific individual the quickest. For graphs of a reasonable size, this kind of graph query is difficult, but for the graphs that underpin today's social networks—the majority of which have billions of edges and millions of nodes—it becomes computationally unfeasible. They suggest Atlas, a cutting-edge method that uses a group of spanning trees to calculate scalable approximate shortest paths between network nodes. In comparison to original graphs, spanning trees are compact, simple to create, and may be dispersed among processors to facilitate query parallelization. Six massive social networks from Facebook, Orkut, and Renren—the largest of which has 43 million nodes and 1 billion edges—are used to illustrate its scalability and efficacy. They outline methods for gradually updating Atlas in response to changes in social graphs over time. Finally, they demonstrate how Atlas may be used to achieve results that are very close to ideal results by applying it to a number of graph applications.

The Travelling Salesman Problem (TSP) was addressed by *S. Goyal et. al [9]* in combinatorial optimisation in both theoretical computer science and operations research. The goal is to discover the shortest route that visits every city precisely once, given a list of cities and their pairwise distances. One of the optimisation issues that has been explored the most, it was first posed as a mathematical problem in 1930. The study of crystal structure, material handling in a warehouse, and data array clustering are the three areas where TSP structural problems most frequently arise.

According to *W. Shalu et. al* [10], researchers have focused on the Cable and Trench Problem in this work, a network design issue that entails balancing capital expenses for network construction against utilisation costs. Building a wider network, or the shortest path tree, may be more expensive, but by having more appealing origin-destination paths, it may lower utilisation costs. On the other hand, lower utilisation costs may result from a smaller network (minimum spanning tree). They give a heuristic that yields optimal or nearly optimal solutions. This heuristic is a modification of the Savings method, which was presented by Clarke and Wright in 1964 to address a vehicle routing issue.

According to *K. M. Curtin et. al [11]*, the network's shape may so naturally depict complex systems that it is an enticing study paradigm. In an increasingly complicated world, it is critical to be able to comprehend the complex systems that surround us, whether they be elaborate communication systems like the internet, transportation networks, or interactions at the cellular level. They give the underlying spatial nature of networks, there is a clear need for network analysis research that could have applications in a variety of academic fields.

A lot of research was done in relation to intricate communication networks by *K A. Ahmat et.al [12]*. They discussed the fundamental ideas of graph theory and how they relate to networks of communication. After that, they are given a few optimisation issues pertaining to network monitoring and routing protocols, and it is demonstrated that a large number of these problems are NP-Complete or NP-Hard. Lastly, they provided an explanation of a few popular tools for creating graph theory-based network topologies.

By taking advantage of rich symmetry in graphs, *Y.Xiao et al [13]* tackles the challenge of online shortest path inquiries. When it comes to priority queues and shortest path queries, Dijkstra is the most

well-known and frequently utilised algorithm for solving the shortest path problem. It is quick and works well with heap data structures.

IV. A STUDY OF GRAPHS METHODS IN NETWORK ROUTE AND ITS ANALYSIS

A system of points with distances between them is called a network. A network can be thought of as pipes, cables, roadways, etc. Determining the shortest path between two points in a network is a common problem with networks. In a shortest path problem, different features (different prices and profits) are typically taken into consideration on numerous real-world occasions. The necessity to create an effective process for addressing these issues arises from the regular occurrence of these network-structured challenges. For this class of problems, there are a few efficient techniques, including the shortest path algorithm. Any node in a network can have edges connecting it to any number of other nodes. The cost of travelling across a link is shown by its cost, weight, distance, and length in the majority of depictions. The best path from one place to another can be found by using the shortest pathways.

1. Dijkstra's Algorithm: It is a fundamental graph search algorithm used to find the shortest path between nodes in a graph with non-negative edge weights.

It works as follows:

- Initialize distances: Assign a tentative distance value to every node. Initially, this value is set to infinity for all nodes except the starting node, which is set to 0.
- Set the initial node as the current node.
- For the current node, consider all of its neighbors and calculate their tentative distances through the current node. Compare these newly calculated tentative distances to the current assigned value and update the tentative distance if it's smaller.
- After considering all neighbors of the current node, mark the current node as visited and remove it from the list of unvisited nodes.
- If the destination node has been visited or if the smallest tentative distance among the nodes in the unvisited set is infinity (indicating that there's no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
- Otherwise, select the unvisited node with the smallest tentative distance, set it as the new current node, and go back to step 3.
- Dijkstra's Algorithm guarantees that once a node has been marked as visited, its distance from the starting node is finalized and optimal.

This algorithm efficiently finds the shortest path from a single source node to all other nodes in the graph, but it doesn't work correctly if the graph contains negative edge weights, as it may produce incorrect results in such cases.

2.Floyd-Warshall's Algorithm: It is another popular algorithm used in graph theory to find the shortest paths between all pairs of vertices in a weighted graph. Unlike Dijkstra's Algorithm, which finds the shortest paths from one source vertex to all other vertices, Floyd-Warshall's Algorithm computes the shortest paths between all pairs of vertices in the graph.

Here's how the algorithm works:

- Initialization: Initialize a 2D array dist[][] where dist[i][j] represents the shortest distance from vertex i to vertex j. Initialize the array with the weights of the edges between vertices if there is an edge, or infinity if there is no edge. Also, initialize the diagonal elements dist[i][i] to 0 since the distance from a vertex to itself is always 0.
- Iterative Update: For each vertex k from 1 to V (where V is the number of vertices), consider all pairs of vertices (i, j). For each pair, check if the path from i to j through k is shorter than the current shortest path from i to j. If it is, update the value of dist[i][j] to the new shorter distance.
    1. The update rule is: dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])
- Termination: After V iterations, the dist[][] array will contain the shortest distances between all pairs of vertices in the graph.

The key advantage of Floyd-Warshall's Algorithm is its ability to handle graphs with negative edge weights (but no negative cycles). It's also relatively simple to implement and understand compared to other algorithms for all-pairs shortest paths.

However, the algorithm's time complexity is O(V^3), where V is the number of vertices in the graph. Therefore, it may not be as efficient as Dijkstra's Algorithm for finding the shortest paths from a single source to all other vertices in sparse graphs.

3.**Bellman-Ford Algorithm**: It is another fundamental algorithm used to find the shortest paths from a single source vertex to all other vertices in a weighted graph, even in the presence of negative edge weights. It's a versatile algorithm that works with graphs that may contain negative edge weights, as long as there are no negative cycles reachable from the source vertex.

Here's how the Bellman-Ford Algorithm works:

- **Initialization**: Initialize an array **dist[]** where **dist[i]** represents the shortest distance from the source vertex to vertex **i**. Initially,0 set **dist[source] = 0** and set all other distances to infinity.
- **Relaxation**: Relaxation is the process of improving the shortest distance to vertices iteratively. Iterate through all edges **u-v** of the graph, and for each edge, if the distance to vertex **v** can be shortened by going through edge **u-v**, update **dist[v]** to the new shorter distance. The update rule is: **dist[v] = min(dist[v], dist[u] + weight(u, v))**, where **weight(u, v)** represents the weight of the edge from **u** to **v**.
- **Iterate**: Repeat step 2 for a total of **V - 1** iterations, where **V** is the number of vertices in the graph. This ensures that after **V - 1** iterations, the shortest paths have been found, as the longest possible shortest path can have at most **V - 1** edges.
- **Check for Negative Cycles**: After **V - 1** iterations, if there are still updates possible, it means that there exists a negative cycle in the graph reachable from the source vertex. The algorithm can be extended to detect negative cycles by performing an additional iteration. If any **dist[v]** can be further improved after **V** iterations, then there is a negative cycle reachable from the source.

The Bellman-Ford Algorithm is less efficient than Dijkstra's Algorithm in the absence of negative edge weights, as its time complexity is O(V * E), where **V** is the number of vertices and **E** is the number of edges. However, it's more versatile in handling negative weights and negative cycles.

4.**Breadth-First Search (BFS):** It is a fundamental graph traversal algorithm used to explore all the vertices in a graph or tree systematically. It starts at a chosen vertex and explores all of its neighbors at the present depth level before moving on to the vertices at the next depth level.

Here's how BFS works:

1. **Initialization**: Choose a starting vertex as the root of the search. If the graph is represented using an adjacency list or matrix, mark all vertices as unvisited.
2. **Explore Neighbors**: Begin by visiting the starting vertex and enqueueing it into a queue data structure. Mark the starting vertex as visited. Then, while the queue is not empty, repeat the following steps:
    - Dequeue a vertex from the front of the queue.
    - Visit and process the dequeued vertex.
    - Enqueue all the unvisited neighbors of the dequeued vertex into the queue.
    - Mark each visited neighbor as visited.
3. **Termination**: Stop when the queue becomes empty, indicating that all reachable vertices have been visited.

BFS systematically explores vertices in a level-by-level manner, ensuring that closer vertices are visited before exploring farther ones. It's often used to find the shortest path in an unweighted graph or to traverse graphs in a level-order manner, such as in tree traversal.

BFS can also be used to find connected components in an undirected graph, determine if a path exists between two vertices, or solve puzzles and problems that involve traversing a graph or tree.

The time complexity of BFS is O(V + E), where V is the number of vertices and E is the number of edges in the graph. This is because BFS visits each vertex and edge exactly once.

5. Johnson's Algorithm: It is a graph algorithm used for finding the shortest paths between all pairs of vertices in a weighted graph, even if the graph contains negative edge weights. It combines elements of Dijkstra's Algorithm for single-source shortest paths and the Bellman-Ford Algorithm for handling negative edge weights.

Here's how Johnson's Algorithm works:

- **Graph Transformation**: First, Johnson's Algorithm transforms the original graph by adding a new vertex and connecting it to all other vertices with zero-weight edges. This transformation ensures that all edge weights become non-negative.
- **Bellman-Ford Algorithm**: Apply the Bellman-Ford Algorithm to the transformed graph starting from the newly added vertex. This step detects negative cycles and updates the distances from the added vertex to all other vertices.
- **Graph Reweighting**: After running the Bellman-Ford Algorithm, reweight the edges of the original graph using the computed distances from the added vertex. This step ensures that all edges have non-negative weights.
- **Dijkstra's Algorithm**: Apply Dijkstra's Algorithm for each vertex in the graph as a source vertex to compute the shortest paths to all other vertices. Since the graph has been reweighted, Dijkstra's Algorithm can be used to find the shortest paths efficiently.
- **Output**: After running Dijkstra's Algorithm for each vertex, the algorithm outputs the shortest path distances between all pairs of vertices.

Johnson's Algorithm is useful for finding shortest paths in graphs with negative edge weights as long as there are no negative cycles reachable from any source vertex. It's a more efficient alternative to Floyd-Warshall's Algorithm for sparse graphs with negative weights since it has a time complexity of O(V * E * log(V)), where V is the number of vertices and E is the number of edges.

However, Johnson's Algorithm may not perform as well as other algorithms for dense graphs due to the additional overhead of graph transformation and the Bellman-Ford step.

V. CONCLUSION

Dijkstra's Algorithm is designed to find the shortest path from a single source vertex to all other vertices in a graph with non-negative edge weights. It is used in network routing protocols, GPS systems, and pathfinding in games. Floyd-Warshall's Algorithm computes the shortest paths between all pairs of vertices in a weighted graph, regardless of negative edge weights. Used in traffic modeling, airline scheduling, and network topology. Bellman-Ford Algorithm finds the shortest path from a single source to all other vertices in a graph, including graphs with negative edge weights, and can detect negative cycles. It is used in network routing protocols, distributed systems, and resource allocation. BFS systematically explores all vertices and edges of a graph level by level, starting from a chosen source vertex. It is used in network analysis, social network analysis, and shortest path problems in unweighted graphs. Johnson's Algorithm efficiently finds all pairs shortest paths in graphs with possibly negative edge weights, avoiding the inefficiency of Floyd-Warshall's Algorithm for sparse graphs with negative weights. This one is used in scenarios where Floyd-Warshall's Algorithm is inefficient due to negative edge weights. In conclusion, each algorithm serves distinct purposes and has its own set of strengths and weaknesses. The choice of algorithm depends on the specific requirements of the problem and the characteristics of the input graph. Understanding these algorithms and their nuances is crucial for selecting the most suitable algorithm for a given problem domain.

REFERENCES

[1]. Biggs, Norman, E. Keith Lloyd, and Robin J. Wilson. *Graph Theory, 1736-1936*. Oxford University Press, 1986.

[2]. Liu, Xiaogang, and Sanming Zhou. "Eigenvalues of Cayley graphs." *arXiv preprint arXiv:1809.09829* (2018).

[3]. Mauldin, R. Daniel, William D. Sudderth, and Stanley C. Williams. "Polya trees and random distributions." *The Annals of Statistics* (1992): 1203-1221.

[4] Estrada, Ernesto. "Graph and network theory." *Glasgow: University of Strathclyde* (2013).

[5]. Konig, Dénes. "Graphs and matrices." *Matematikai és Fizikai Lapok* 38 (1931): 116-119.

[6] Harary, Frank, and Robert Z. Norman. *Graph theory as a mathematical model in social science*. No. 2. Ann Arbor: University of Michigan, Institute for Social Research, 1953.

[7] Zhan, F. Benjamin. "Three fastest shortest path algorithms on real road networks: Data structures and procedures." *Journal of geographic information and decision analysis* 1.1 (1997): 69-82.

[8] Cao, Lili, et al. "Atlas: Approximating shortest paths in social graphs." *Computer Science Department* (2011).

[9] Goyal, Sanchit. "A survey on travelling salesman problem." *Midwest instruction and computing symposium*. 2010.

[10] Wadhwa, Shalu. *Analysis of a network design problem*. Diss. Lehigh University, 2000.
[11] Curtin, Kevin M. "Network analysis in geographic information science: Review, assessment, and projections." *Cartography and geographic information science* 34.2 (2007): 103-111.
[12] Ahmat, Kamal. "Graph Theory and Optimization Problems for Very Large Networks." *arXiv preprint arXiv:0907.3099* (2009).
[13] Xiao, Yanghua, et al. "Efficiently indexing shortest paths by exploiting symmetry in graphs." *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*. 2009.
[14] Eisenstein, Daniel J., et al. "SDSS-III: Massive spectroscopic surveys of the distant universe, the Milky Way, and extra-solar planetary systems." *The Astronomical Journal* 142.3 (2011): 72.
[15] Borgwardt, Karsten M., and Hans-Peter Kriegel. "Shortest-path kernels on graphs." *Fifth IEEE international conference on data mining (ICDM'05)*. IEEE, 2005.
[16] Pallottino, Stefano, and Maria Grazia Scutella. "Shortest path algorithms in transportation models: classical and innovative aspects." *Equilibrium and advanced transportation modelling*. Boston, MA: Springer US, 1998. 245-281.
[17]N.K. Kuppuchamy, and R. Manimegalai, "Manet Routing: Optimization by Genetic and Fuzzy Logic Approach," Journal of Theoretical and Applied Information Technology, 2013.
[18] H. Nazif, and L.S. Lee, "Optimized Crossover Genetic Algorithm for Vehicle Routing Problem with Time Windows" American Journal of Applied Sciences, 2010, 7(1) pp. 95-101.
[19] R. Nallusamy, K. Duraiswamy, R. Dhanalaksmi, and P. Parthiban, "Optimization of multiple Vehicle Routing Problems using Approximation Algorithms," International Journal of Engineering Science and Technology, 2009 vol 1 (3).
[20] Shirinivas, S. G., S. Vetrivel, and N. M. Elango. "Applications of graph theory in computer science an overview." *International journal of engineering science and technology* 2.9 (2010): 4610-4621.