International Journal of Mathematics and Statistics Invention (IJMSI)
e-ISSN: 2321 – 4767 P-ISSN: 2321 - 4759
www.ijmsi.org Volume 13 Issue 3 || May - June 2025 || PP: 10-18

AN ANALYTICAL FRAMEWORK OF SHORTEST PATH OPTIMIZATION USING GRAPH THEORY

Mrs. R. Devi and R. Krishnaloshini***

****Assistant Professor, PG Department of Mathematics, Bon Secours Arts and Science College for Women, Mannargudi, India.**

***Master of Science, PG Department of Mathematics, Bon Secours Arts and Science College for Women, Mannargudi, India**

Date of Submission: 05-05-2025 Date of acceptance: 18-05-2025

ABSTRACT

This project presents an analytical framework for shortest path optimization using graph theory, with a focus on real-world applications in logistics, transportation, telecommunications, and finance. By modeling complex systems as graphs, we explore how graph-theoretic algorithms can enhance decision-making and operational efficiency. The study primarily investigates two widely used shortest path algorithms such as Dijkstra's Algorithm and the Bellman-Ford Algorithm. We evaluate their performance in terms of computational efficiency, accuracy, and applicability to various network conditions, including graphs with negative edge weights. Through a series of experiments and real-world case studies, we demonstrate how these algorithms can be effectively applied to optimize routes and minimize travel distances. The findings provide comparative insights into the strengths and limitations of each approach, offering practical guidance on selecting the appropriate algorithm based on specific problem constraints. This research underscores the critical role of graph theory in solving optimization problems and improving network-based decision-making.

KEYWORDS: Shortest path optimization, Graph theory, Transportation, Telecommunications, Finance, Graph-theoretic Algorithms, Dijkstra's Algorithm, Bellman-Ford Algorithm, Optimize routes, Minimize Travel distances, Network-based Decision-making.

1. INTRODUCTION

Optimization challenges are central to many modern industries, including logistics, transportation, telecommunications, and finance where efficient decision-making is critical to operational success. As problems in these domains grow increasingly complex and large-scale, traditional optimization methods often fall short in delivering scalable and effective solutions. Graph theory, with its mathematical foundation for modeling relationships between entities, offers a powerful framework to tackle such challenges.

This paper focuses on the intersection of graph theory and optimization modeling, particularly in the context of shortest path algorithms. By representing systems as graphs, we can capture complex interdependencies and apply structured approaches to decision-making. Graph-based models enable the development of efficient algorithms that improve routing, reduce operational costs, and enhance performance across a range of real-world applications.

We explore key graph-theoretic techniques especially shortest path algorithms such as Dijkstra's Algorithm and the Bellman-Ford Algorithm and analyze their practical relevance in solving optimization problems. From optimizing delivery routes in logistics to managing data flow in telecommunications networks, these algorithms offer scalable and precise solutions for dynamic, real-time environments.

2. DEFINITIONS

2.1. OPTIMIZATION MODELING:

An optimization modeling is a mathematical tool, that is used to determine the most optimal solution to a problem through the examination of constraints and objectives. It is utilized to assist people and groups in making well-informed choices by optimizing or minimising a specific objectives while following particular limitations.

2.2. GRAPH THEORY:

Graph theory is the branch of mathematics, which is the study of structures made up of nodes connected by links, used to represent relationships and networks.

2.3. GRAPH:

A graph (denoted as $G = (V, E)$) consists of a non-empty set of vertices V and a set of edges E. A vertex a represents an endpoint of an entities. An edge e joins two vertices a, b and is represents the relationship.

Where $V = \{a, b, c, d\}$ and $E = \{e, f, g, h\}$

2.4. PATH:

A sequence of edges that connects a sequence of vertices in a graph is said to be path.

2.5. WEIGHTED GRAPH:

A weighted graph is a graph in which each edge is assigned a numerical value, called a weight. These weights can represent various metrics, such as distance, cost, or time, depending on the context.

2.6. DIRECTED GRAPH

A directed graph is a graph in which the edges may only be traversed in one direction. Edges in a simple directed graph may be specified by an ordered pair (v_i, v_j) of the two vertices that the edge connects. We say that v_i is adjacent to v_j and v_j is adjacent from v_i .

2.7 SHORTEST PATH ALGORITHM

In shortest path routine the path length between each node is measured as a function of distance, bandwidth, average traffic, communication cost, measured delay etc, in a graph The node representing a router and arc of the graph representing a communication link each link has a cost associated with it. To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.

3. DIJKSTRA'S ALGORITHM

Dijkstra's algorithm is a popular algorithm for solving many single source shortest path problems having non-negative edge weight in the graphs. i.e., it is to find the shortest distance between two vertices on a graph. This algorithm is called single source shortest path because in this algorithm, for a given vertex called source. The shortest path to all other vertices is obtained.

3.1. STEPS OF DIJKSTRA'S ALGORITHM

STEP 1 – Set the starting node (Source node):

Set the starting node's distance to zero and mark it as current node and the rest with infinity.

STEP 2 — Calculate distances:

Find all vertices that lead to the current node and calculate their distances.

STEP 3 — Mark the current node as visited:

Once all the neighbours of the current node have been visited, mark it as visited.

STEP 4 — Find the next current node:

Choose the unvisited node with the smallest distance as the current node.

STEP 5 — Repeat :

Repeat the process until the shortest distance to the end node is found.

3.3. FINDING SINGLE SOURCE SHORTEST PATH USING DIJKSTRA'S ALGORITHM

The steps involved in Dijkstra's Algorithm to find the single source shortest distance.

- **Step 1:** Mark the source node with a current distance of 0 and the rest with infinity.
- **Step 2:** Set the non-negative node with the smallest current distance as the current node.
- **Step 3:** For each neighbour(N) of the current node adds the current distance of the adjacent node with the weight of the edge connecting $0 \rightarrow 1$. If it is smaller than the current distance of node, set it as new current distance N.
- **Step 4:** Mark the current node 1 as visited.
- **Step 5:** If there are any nodes unvisited, repeat the process from step 2.

4. BELLMAN-FORD ALGORITHM

Bellman–ford algorithm is the extension of Dijkstra's Algorithm that can be handle negative weight edges. It works by relaxing the edges of the graph repeatedly.

4.1. STEPS OF BELLMAN-FORD ALGORITHM

STEP 1 — Initialization:

Set the distance of the source vertex to 0 and all the vertices to infinity. Also, set the predecessor of all the vertices to null.

STEP 2 — Relaxation:

Repeat this step for each edge in the graph $|V|$ - 1 times. During each iteration, update the distance to each vertex if there is a shorter path through another vertex.

STEP 3 - Check for negative cycles:

Perform one final iteration to check for negative cycles. If a shorter path can still be found, then there is a negative cycle in the graph.

STEP 4- Output the shortest paths:

Return the shortest distances and predecessors.

4.2. FINDING SHORTEST PATH USING BELLMAN-FORD ALGORITHM

The steps involved in bellman-ford Algorithm to find the single source shortest distance.

- **Step 1:** Set the distance of the source vertex to 0 and all the vertices to infinity.

- **Step 2:** Repeat this step for each edge in the graph |V| - 1 times, where |V| is the number of vertices in the graph. For each edges (u,v) with weight W, check if the distance from the source vertex to v can be reduced by going through u. If so, update the distance to v to the new, shorter distance.
- **Step 3:** Check the negative weight cycles. If there is a negative weight cycle in the graph, the algorithm will never converge and will keep reducing the distance to some vertices with each iteration. To detect such cycles, repeat step to one more time. If any distance is updated in this extra iteration, there must be a negative weight cycle in the graph.
- **Step 4 :** If there is no negative weight cycle, the shortest distance to each vertex from the source vertex has been found.

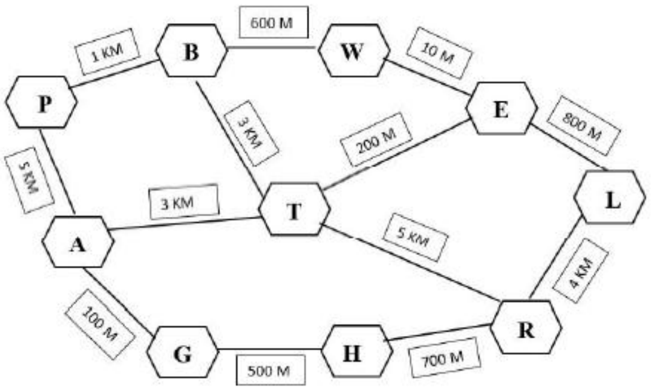
5. DIFFERENCE BETWEEEN SHORTEST PATH ALGORITHMS

SHORTEST PATH ALGORITHM	
DIJKSTRA'S ALGORITHM	**BELLMAN-FORD ALGORITHM**
➢ It is greedy algorithm.	➢ It is dynamic programming algorithm.
➢ All edge weight must be non-negative.	➢ Can handle negative edge weights, but may not work correctly if there are negative weight cycles.
➢ Perform in the weighted and unweighted graphs.	➢ Perform in the weighted and directed graph.
➢ It is efficient for finding shortest paths from a single source.	➢ Suitable for graphs with negative edge weights or when negative weight cycles need to be detected.

6. ALGORITHM BASED PROBLEM

Determine the shortest path to help the person to see the lion in a weighted graph representing the zoological park.

The graph is



The graph G(S,D) where S are the nodes and D are the arcs which contains 10 nodes and 13 arcs.

Here the nodes are spots in the zoological park and the arcs are distance from one spot to another in it.

Where the nodes are {P, B, A, W, E, L, R, H, T, G}

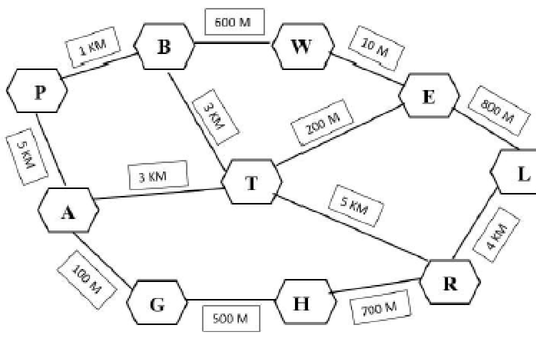
- ✓ P – Person
- ✓ B – Birds
- ✓ A – Aquarium
- ✓ W – Wolf
- ✓ E – Elephant
- ✓ L – Lion
- ✓ R – Rhinoceros
- ✓ H – Hippopotamus
- ✓ T – Tiger
- ✓ G – Giraffe

And the edges are { (P, B), (P, A), (B, W), (B, T), (W, E), (E, L), (E, T), (L, R), (R, T), (R, H), (H, G), (G, A), (A, T) }.

SOLUTION

DIJKSTRA'S ALGORITHM

Let the given graph be,



STEP 1: Mark the source node with a current distance of 0 and the rest with infinity.

Therefore, select the node P is initialised as source node.

Write the distance from source node to other node.

S = { P }

H = {B, A, W, E, L, R, H, T, G}

- ✓ D (P, B) = 1 KM (min)
- ✓ D (P, A) = 5 KM
- ✓ D (P, W) = ∞
- ✓ D (P, E) = ∞
- ✓ D (P, L) =∞
- ✓ D (P, R) = ∞
- ✓ D (P, H) = ∞
- ✓ D (P, T) = ∞
- ✓ D (P, G) = ∞

➢ Here D stands for distance

STEP 2: Set the non-negative node with the smallest current distance as the current node.

In the above distance, D (P, B) is the minimum weight which is 1 KM.

Therefore choose the node B.

STEP 3: For each neighbour(N) of the current node adds the current distance of the adjacent node with the weight of the edge connecting $0 \rightarrow 1$. If it is smaller than the current distance of node, set it as the new current distance N.

Hence, the path is $P \rightarrow B$

STEP 4: Mark the current node B as visited.

STEP 5: If there are any nodes are unvisited, repeat the process from step 2.

Here there are more unvisited nodes. So, we repeat the process from step 2. The current node are

S = {P, B}

H = { A, W, E, L, R, H, T, G}

- ✓ D (P, A) = 5 KM
- ✓ D (P, W) = 1 KM 600M
- ✓ D (P, T) = 4 KM
- ✓ D (P, E) = ∞
- ✓ D (P, L) =∞
- ✓ D (P, R) = ∞
- ✓ D (P, H) = ∞
- ✓ D (P, G) = ∞

Here the node W be the new current node.

Therefore the path becomes $P \rightarrow B \rightarrow W$

The current node are

S = {P, B, W}

H = { A, E, L, R, H, T, G}

- ✓ D (P, A) = 5 KM
- ✓ D (P, T) = 4 KM
- ✓ D (P, E) = 1 KM 610 M
- ✓ D (P, L) =∞
- ✓ D (P, R) = ∞
- ✓ D (P, H) = ∞
- ✓ D (P, G) = ∞

Here the minimum distance is 1 KM 610 M, therefore we connect the path as

$$P \rightarrow B \rightarrow W \rightarrow E$$

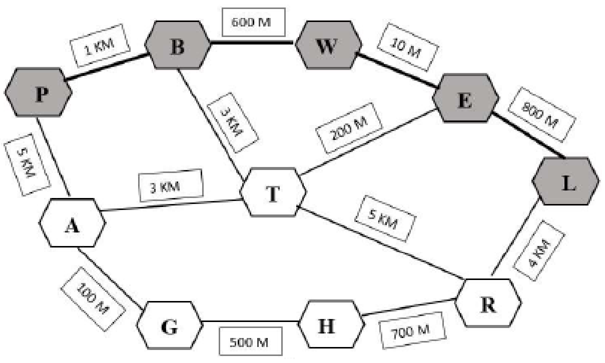
The current node are

S = {P, B, W, E} & H = { A, L, R, H, T, G}

- ✓ D (P, A) = 5 KM
- ✓ D (P, T) = 4 KM
- ✓ D (P, L) = 2KM 410M
- ✓ D (P, R) = ∞
- ✓ D (P, H) = ∞
- ✓ D (P, G) = ∞

Here the minimum distance is 2 KM 410 M, therefore we connect the path as

$$P \rightarrow B \rightarrow W \rightarrow E \rightarrow L$$



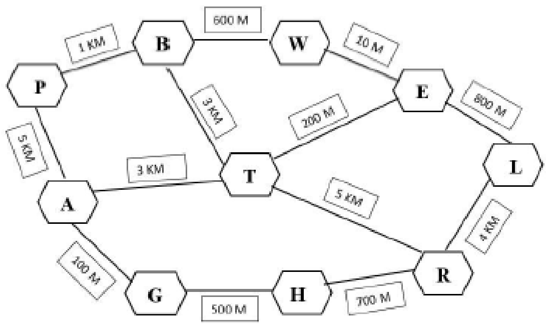
Therefore, we reach the destination, which is Person to Lion.

The shortest path is $P \rightarrow B \rightarrow W \rightarrow E \rightarrow L$

Whose total distance is 2KM 410M Which is the shortest distance.

BELLMAN-FORD ALGORITHM

The given graph is



STEP 1: Set the distance of the source vertex to 0 and all the vertices to infinity.

Here, Node P is initialized as source node.

STEP 2: Repeat this step for each edge in the graph |V| - 1 times, where |V| is the number of vertices in the graph. For each edges (u,v) with weight W, check if the distance from the source vertex to v can be reduced by going through u. If so, update the distance to v to the new, shorter distance.

Distance PB is shorter than the PA, so we chosen the route PB

$$d(PB) < d(PA)$$

The distances are

- ✓ d(PB) = 1 KM
- ✓ d(PW) =1 KM 600M
- ✓ d(PT) = 4 KM

$d(PW) < d(PT)$

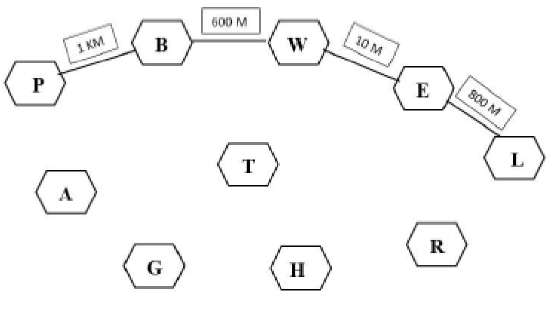
So, d(PW) is chosen.

STEP 3: Check the negative weight cycles. If there is a negative weight cycle in the graph, the algorithm will never converge and will keep reducing the distance to some vertices with each iteration. To detect such cycles, repeat step to one more time. If any distance is updated in this extra iteration, there must be a negative weight cycle in the graph.
In the given graph there is no negative edges. Hence it has no negative cycle.

STEP 4 : If there is no negative weight cycle, the shortest distance to each vertex from the source vertex has been found.

The node W an only one adjacent node, hence we chosen d(PE).

Likewise, for the node E, it also has an only one adjacent node,



Therefore the shortest path is **PBWEL** Which has the shortest distance 2KM 410M.

From this two algorithm, we conclude that, to finding the shortest path, **Dijkstra's algorithm** is generally considered better than Bellman-Ford, as it is significantly faster, especially when dealing with graphs with non-negative edge weights; however, Bellman-Ford is the preferred choice if your graph contains negative edge weights because Dijkstra cannot handle them properly.

7. CONCLUSION

Through analytical evaluation and real-world testing, this study demonstrates the effectiveness of graph theory in solving shortest path problems across various domains. By comparing the performance of Dijkstra's Algorithm and the Bellman-Ford Algorithm, we found that Dijkstra's Algorithm consistently outperforms in terms of speed and efficiency on graphs with non-negative edge weights, making it highly suitable for most practical applications such as transportation and logistics. While the Bellman-Ford Algorithm remains valuable for handling graphs with negative weights, Dijkstra's Algorithm proves to be the optimal choice when negative weights are not a concern, due to its faster execution and lower computational complexity. Overall, our findings affirm that graph-theoretic approaches, particularly Dijkstra's Algorithm, are powerful tools for enhancing optimization and decision-making in complex network environments.

8. BIBLIOGRAPHY

[1] *Douglas B. West*, "Introduction to graph theory" (2nd edition).

[2] *Frederick S. Hillier, Gerald J. Lieberman* (2015), "Introduction to Operations Research", (10th Edition).

[3] *Jonathan L. Gross, Jay Yellen(*2005), "Graph Theory and Its Applications", (3rd Edition).

[4] *Ravindra K. Ahuja, Thomas L. Magnanti, James B. Orlin* (1993), "Network Flows: Theory, Algorithms, and Applications", (2nd Edition).

[5] *Robert Sedgewick, Kevin Wayne* (2011), "Graph Algorithms".