RT&A, No 1 (82)
Volume 20, March 2025

A MODIFIED INCIDENT EDGE PATH ALGORITHM FOR EFFICIENT SHORTEST PATH SOLUTIONS IN PIPELINE NETWORKS AND URBAN NAVIGATION SYSTEMS

Kanchana M[1] and Kavitha K*

•

Department of Mathematics, Vellore Institute of Technology, India
[1]mkanchana378@gmail.com and *kavinphd@gmail.com

Abstract

The article describes how to utilize the Modified Incident Edge Path Algorithm (MIEPA) to identify the cheapest transit option and the best route. The MIEPA algorithm, which is based on graph theory, is simple to use and can potentially be employed to major smart logistics challenges such as pipeline networks and Google Maps. It evaluates the most optimal approach to minimize transportation expenses using MATLAB. The algorithm ensures that each node gets visited and determines the shortest path from the origin to all other nodes. The running time complexity and theorem of the new method are presented, and the algorithm is compared to the existing algorithm. The proposed MIEPA addresses negative weights and prevents negative cycles. It has used two real-world problems to evaluate the suggested algorithm.

Keywords: Directed network, Incident Edge Path Algorithm, Optimum path, Pipeline networks, Google Map Network, Minimum transportation cost.
AMS Subject Classification: 05C85, 94C15, 90C27, 90C35

I. Introduction

Network optimization is a vital topic of operations research and smart logistics, with the goal of identifying the most effective routes while decreasing transportation costs. Network models are primarily concerned with addressing the Shortest Path Problem (SPP), which involves establishing the optimal path between nodes in a variety of real-world settings. This challenge is essential in various disciplines, including:

- Manufacturing: Determining the shortest path simplifies operations, resulting in faster and more productive manufacturing.
- Management: Rational thinking is critical for maximizing benefits and minimizing expenses.
- Transportation: Designing effective routes is critical for moving big amounts of commodities at a low cost.

In essence, the shortest path problem covers multiple challenges, aiming to maximize outcomes in manufacturing, top management, and transportation. The basic definitions, terminologies and notations we refer[4, 5, 6].Many shortest path algorithms in graph theory, including Dijkstra's, Prim's, Kruskal's, and other shortest path and minimum spanning tree algorithms, provide effective solutions for finding optimal paths based on various real-world network problems [2, 10, 11, 12]. Dijkstra's algorithm gives the N-Shortest paths for the network, and it is implemented in Google Map Network problem[9]. A review paper helps the researchers to study well about the algorithms and also for implementation in real-world problems[7]. These algorithms assist decision-makers in identifying the most efficient routes with minimal transportation costs to

reach their destinations.These methods address key operational challenges, such as pipeline maintenance, which can lead to traffic congestion and disrupt the timely delivery of supplies to various businesses. By adopting these strategies, industries can identify the fastest routes with minimal travel time, reducing disruptions and ensuring swift access to affected areas [1, 13]. In addition, optimizing algorithms for both space and time complexity can substantially enhance business profitability. The temporal complexity of greedy algorithms has been thoroughly examined by Yogesh Chanchal and Dr. Ashedra Kumar Saxena (2015), highlighting their role in improving operational efficiency. Refining these algorithms enables industries to manage resources more effectively, lower costs, and increase revenues [16]. Furthermore, Chu Fei-Xui et al. applied the well-established Dijkstra method to calculate the N-shortest paths in pipeline networks [3]. Shortest path methods in graph theory are vital not only for financial applications, but also for network systems, engineering, and computer science. These methods are now widely used in Computer Science and Engineering, highlighting their significance in improving network performance, system design, and computing efficiency[8].
Recently, particularly during the epidemic, there has been a rise in demand for prompt and dependable delivery services. To ensure timely and efficient delivery of food, medicine, and other supplies, delivery agents or organizations must find the shortest routes while simultaneously minimizing fuel expenses. Yassine Issaoui and colleagues addressed these needs in their work, focusing on improving delivery routes to ensure timely deliveries at minimal costs[14]. Jijun Hu and colleagues investigated the use of Dijkstra's Algorithm to calculate the best path for Automatic Guided Vehicles (AGVs) in warehouse distribution systems. Their job entails synchronizing duties with vehicle movements to improve efficiency. They proposed an effective architecture using Dijkstra's Algorithm to address numerous warehouse-related difficulties, hence boosting overall system performance[15]. In this paper, we introduce the Modified Incident Edge Path approach (MIEPA), a unique approach for determining the shortest paths between nodes. The proposed MIEPA is modified from the incident edge path algorithm proposed in[17]. We used MIEPA to solve two real-world problems: (1) determining the best path in a pipeline network to reach pump stations from a specified origin pump station and (2) calculating the shortest path in Google Maps for efficient navigation. The MIEPA algorithm's performance in these applications is examined for reference [3].Novelty of the proposed MIEPA is it handles negative weighted edges of the transportation network by avoiding cycles unlike existing algorithms. It is appropriate for sparse networks because it streamlines the process of determining the minimal or most effective route from the origin to all subsequent nodes without requiring backtracking. This paper is structured as follows:
Introduction: This article introduces the Modified Incident Edge Path Algorithm (MIEPA), its concepts, and a theoretical proof.
Application: The algorithm is evaluated in real-life applications, including pipeline networks and navigation systems.
Evaluation: We assess the effectiveness of the MIEPA algorithm and discuss how it addresses its limitations and shortcomings.

II. Preliminaries

A graph $G(V, E)$ is directed and taken as a network representation, where V is the number of nodes denoted $|V| = n$ and E is the number of edges denoted $|E| = m$. In G, the node v_i is an end node of some edge e_j, v_i and e_j is said to be incident with each other. The edges incident with v_i are said to be incident edges of G [5]. The distance between nodes v_i and v_j is $d(v_i, v_j)$. The weight of the node v is considered as the cost of each u, v-path. If v has more than one u, v-path, then the minimum cost is assigned to the end node v.
A graph having no loops or parallel edges is a simple network [5]. A path is a simple network whose nodes can be linearly ordered so that two nodes are adjacent if and only if they are one after the other in the list [4]. A path $(v_i - v_j)$ from the node v_i to v_j in a network is finite if it has a finite number of nodes; otherwise, it is infinite. The weight of a path is taken as the sum of the

distance between nodes in the respective path. If the path starts and ends with the same node, then it is known as a cyclic, otherwise acyclic, of the network [10].
Result: Let $G(V, E)$ be a simple and directed network; if G has only one shortest path from the origin node v_0 to the destination node v_n, then it is the optimum path for the network, providing minimum transportation cost. The proposed algorithm was examined by Kanchana et al. [17], this article presents the modification of IEPA and applies it in the pipeline network and Google Map Service network to find the shortest path of every node from the origin node. In the Modified Incident Edge Path Algorithm (MIEPA), we make the following assumptions:

- Taking the graph as a directed network (graph).
- The distance, costs, and time needed are associated with edges of the network that travels
- Single source with negative or non-negative edge weights avoiding cycles.
- Optimality of path drawn from each node from the origin node.
- The proposed algorithm involves performing optimality without backtracking from the beginning.

I. Modified Incident Edge Path Algorithm (MIEPA)

Formulation of Graphical problem: $G(V, E)$ be the directed simple graph. Each edge has the distance between the nodes.

- v_0 be the origin node and v_n be the end node of the graph.
- $V-$ Set of all nodes
- $E-$ Set of all edges/lines between nodes
- $d(v_i, v_j)-$ distance between the nodes
- E_i- Set of incident edges where $i = 1, 2, .., n$ and$j = 1, 2, .., n-1$.

A network $G(V, E)$, with node set V and edge set E, we set p_j the weight of the node v_i and E_j, the set of all incident edges of G where $0 \leq i \leq n, 1 \leq j \leq n$.
The following theorem provides the optimality of the proposed algorithm.

Theorem 1. Let $G(V, E, d(v_i, v_j))$ be any simple directed graph and the weight of the path $p_j = min\{p_i + d(v_i, v_j)\}$ is defined on G. If $E_n = \{\}$ then, the weight p_n of the path $(v_0 - v_n)$ is optimum, Where $d(v_i, v_j)$is the distance between nodes v_i and v_j for $i = 0, 1, ..., n, j = 1, 2, , n$.

Proof.
Let $G(V, E, d(v_i, v_j))$ is any simple directed graph with vertex set $V = \{v_0, v_1, ..., v_i, v_j, .., v_n\}$ and E_j is the collection of all incident edges incident with v_i of G.
given that, $p_j = min\{p_i + d(v_i, v_j)\}$ as the weight of path from v_i to v_j. Let E_1 is the set of incident edges from v_0 vertex, then the weight of path $v_0 - v_1$ is $p_1 = p_0 + d(v_0, v_1)$ E_2 is the set of incident edges from v_1, then the weight of path $(v_0 - v_1 - v_2)$ is $p_2 = p_1 + d(v_1, v_2)$,
If a vertex v_1 has two in-degree, then choose the minimum weight of the path with corresponding path. Thus, for v_j vertex, E_j is the set of incident edges, then the weight of the path $(v_0 - v_1 ... v_i - v_j)$ is $p_j = min\{p_i + d(v_i, v_j)\}$. Similarly for vertex v_n, E_n is empty since there are no edges incident with v_n. Thus $(v_0 - v_1 ... v_i - v_j ... v_{n-1} - v_n)$ is the obtained path from $p_n = min\{p_{(n-1)} + d(v_{(n-1)}, v_n)\}$ which is optimum.

■

Remark 1: If any nodes in-degree is more than one then we can choose and fix the minimum weight path as $p_j = min\{p_i + d(v_i, v_j)\}$ of the nodes from v_i to v_j and also fix the path weight as of the respective end nodes weight.
Remark 2: If any node has more than one weight, choose the minimum weight path and fix that minimum path weight as the end node's weight.

II. MIEPA - Steps

- Step 1: Find E_1, the set of all edges incident with v_0 in V. If E_1 is empty, go to Step 6 or proceed to Step 2.

- Step 2: Calculate path weight from E_1, $p(v_0, v_j) = w_0 + d(v_0, v_j)$. Fix the weight to end node of respective and choose the minimum without forming a cycle.
- Step 3: Find E_2, the set of all edges incident with v_j in V. If E_2 is empty, go to Step 6 or go to Step 4.
- Step 4: Calculate path weight from E_2, $p(v_j, v_i) = w_j + d(v_j, v_i)$. Fix the weight to the end node of the respective and choose the minimum without forming a cycle.
- Step 5: Repeat Steps 1 to 4 until the destination node is reached after visiting all edges. Optimum cost is the weight of the destination node.
- Step 6: Check for any negative weight cycles. Check for any weight adjustments; if a negative cycle exists or the optimal path is obtained, stop the process.

III. Comparison of MIEPA with existing algorithms

The Table 1, provides proposed MIEPA's weightage compared with other existing algorithms.

Table 1: *Comparison of Proposed MIEPA and Other Existing Algorithms*

Aspect	Proposed MIEPA	Dijkstra's Algorithm	Bellman-Ford Algorithm	A* Algorithm
Graph Type	Focuses on incident edges in dynamic, sparse graphs.	Functions with weighted graphs without negative edges.	Identifies negative cycles and manages negative edge weights.	Works with heuristic-based approaches on graphs.
Efficiency	More effective for sparse graphs as it focuses on local incident edges.	Efficient for dense graphs using priority queues ($O(V \log V)$).	Global edge checks result in slower runtime ($O(VE)$).	Very efficient when appropriate heuristics are available ($O(E)$ to $O(E + V \log V)$).
Time Complexity	$O(V + E)$, dependent on the number of incident edges, ideal for sparse graphs.	$O(V^2)$, or $O(E + V \log V)$ with priority queues.	$O(VE)$, due to global edge assessments.	$O(E)$ to $O(E + V \log V)$ depending on heuristic use.
Negative Weight Handling	Can handle negative weights by identifying and avoiding negative cycles.	Cannot handle negative weights; requires non-negative edges.	Handles negative weights but requires additional processing.	Cannot directly handle negative weights without modifications.
Use Case	Ideal for dynamic, adaptive, local exploration (e.g., transportation networks and pipelines).	Suitable for non-negative edge graphs (e.g., shortest paths in maps).	Useful for graphs with negative weights (e.g., finance, transportation).	Best suited for heuristic pathfinding in domains like games and robotics.
Edge Handling	Concentrates on incident edges (examines neighbors step-by-step).	Evaluates all edges connecting each node.	Evaluates all edges, even for negative weight cycles.	Prioritizes edges using heuristic rankings.
Adaptability	Highly adaptive for local connectivity and dynamic systems requiring cycle detection.	Not suitable for dynamic graphs or graphs with negative weights.	Supports static graphs with negative weights.	Performs well in domains with applicable heuristics (e.g., real-time pathfinding).

III. Smart Logistic Network Problems

I. Pipeline Network Problem

The MIEPA algorithm is used to determine the most cost-effective path in a pipeline network, helping industries like the South-West products pipeline industry optimize their operations. By representing pump stations as nodes and pipelines as edges and calculating the minimum cost using unit costs, the MIEPA algorithm provides an efficient method to find the optimal transportation routes within the network.

Assume Figure 1 illustrates a network diagram with nodes and edges representing pump stations and pipelines, respectively. The MIEPA algorithm will calculate the minimum cost of reaching the destination node by iteratively finding the shortest path in terms of transportation cost. The distance of edges in the pipeline network is shown in Table 2.

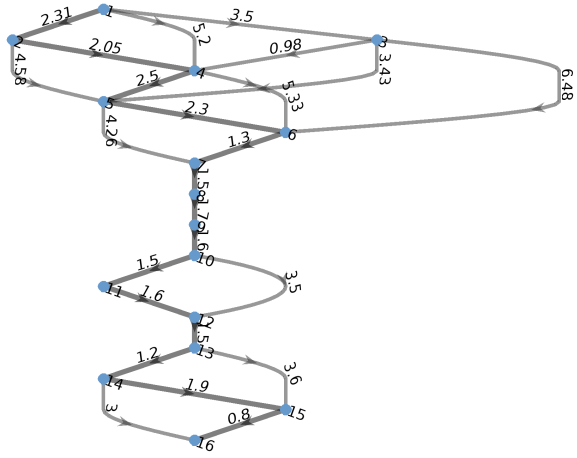

Figure 1: *Network of Pipeline*

Table 2: *Edge and Distance Between the Nodes*

Edge	**Distance between v_i and v_j**	**Edge**	**Distance between v_i and v_j**
(v_0, v_1)	2.31	(v_6, v_7)	1.5
(v_0, v_2)	3.5	(v_7, v_8)	1.7
(v_0, v_3)	5.2	(v_8, v_9)	1.6
(v_1, v_3)	2.05	(v_9, v_{10})	1.5
(v_1, v_4)	4.58	(v_9, v_{11})	3.5
(v_2, v_3)	0.98	(v_{10}, v_{11})	1.6
(v_2, v_4)	3.43	(v_{11}, v_{12})	1.5
(v_2, v_5)	6.48	(v_{12}, v_{13})	1.2
(v_3, v_4)	2.5	(v_{12}, v_{14})	3.6
(v_3, v_5)	5.33	(v_{13}, v_{14})	1.9
(v_4, v_5)	2.3	(v_{13}, v_{15})	3.0
(v_4, v_6)	4.26	(v_{14}, v_{15})	0.8
(v_5, v_6)	1.3		

The proposed MIEPA algorithm is applied, and the calculation for getting the optimum path is given in Table 3 for the pipeline network.

Table 3: *Pipeline Network Problem Solved by MIEPA*

Incident Edge Set E_j	**The Weight and the Shortest Path p_j of (v_j)**
$E_1 = \{v_0v_1, v_0v_2, v_0v_3\}$	$p_1 = 2.31$ of (v_1), $p_2 = 3.5$ of (v_2), and $p_3 = 5.2$ of (v_3)
$E_2 = \{v_1v_3, v_1v_4\}$	$p_3 = \min\{5.2, 4.36\} = 4.36$ of (v_3), $p_4 = 6.89$ of (v_4)
$E_3 = \{v_3v_4, v_3v_5\}$	$p_4 = \min\{6.89, 6.86\} = 6.86$ of (v_4), $p_5 = 10.84$ of (v_5)
$E_4 = \{v_4v_5, v_4v_6\}$	$p_5 = \min\{9.16, 10.84\} = 9.16$ of (v_5), $p_6 = 11.09$ of (v_6)
$E_5 = \{v_5v_6\}$	$p_6 = \min\{11.09, 10.46\} = 10.46$ of (v_6)
$E_6 = \{v_6v_7\}$	$p_7 = 11.96$ of (v_7)
$E_7 = \{v_7v_8\}$	$p_8 = 13.66$ of (v_8)
$E_8 = \{v_8v_9\}$	$p_9 = 15.26$ of (v_9)
$E_9 = \{v_9v_{10}, v_9v_{11}\}$	$p_{10} = 16.76$ of (v_{10}), $p_{11} = 18.76$ of (v_{11})
$E_{10} = \{v_{10}v_{11}\}$	$p_{11} = \min\{18.76, 18.36\} = 18.36$ of (v_{11})
$E_{11} = \{v_{11}v_{12}\}$	$p_{12} = 19.86$ of (v_{12})
$E_{12} = \{v_{12}v_{13}, v_{12}v_{14}\}$	$p_{13} = 21.06$ of (v_{13}), $p_{14} = 23.46$ of (v_{14})
$E_{13} = \{v_{13}v_{14}, v_{13}v_{15}\}$	$p_{14} = \min\{22.96, 23.46\} = 22.96$ of (v_{14}), $p_{15} = 24.06$ of (v_{15})
$E_{14} = \{v_{14}v_{15}\}$	$p_{15} = \min\{23.76, 24.06\} = 23.76$ of (v_{15})
$E_{15} = \{\}$	Stop the process; destination reached.

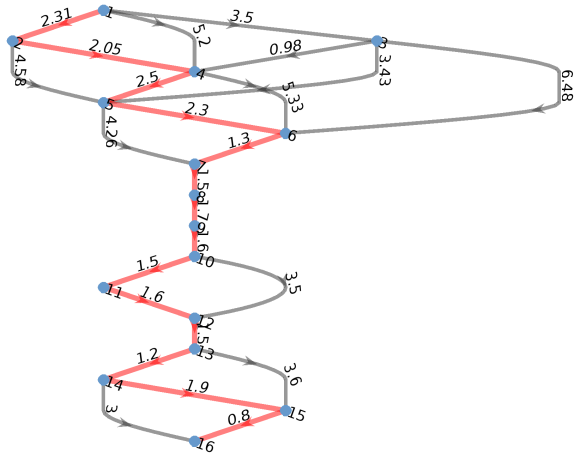

Figure 2: *Network of Pipeline with optimal path by MATLAB*

End node reached stop the process, since $E_{15} = \{\}$ there is no edges incident with v_{15} weight of the end node of selected minimum weight path is $p_{15} = 23.76$. Hence, the minimum transportation cost of the optimum path is 23.76 hundred million yuan.
The Table 4 is a representation of the minimum cost to reach every node v_0 and also the optimum path to reach the end node where all the edges have been visited. Therefore, the optimum path is $(v_0 - v_1 - v_3 - v_4 - v_5 - v_6 - v_7 - v_8 - v_9 - v_{10} - v_{11} - v_{12} - v_{13} - v_{14} - v_{15})$ with (weight of the path) minimum transportation cost $p_{15} = 23.76$ hundred million yuan.

Table 4: *Shortest Path for Each Node from v_0 in the Pipeline Network*

End Node	**Shortest Path** $p(v_0, v_j)$	**Minimum Cost** w_j **of** $p(v_0, v_j)$
v_1	$v_0 - v_1$	2.31
v_2	$v_0 - v_2$	3.5
v_3	$v_0 - v_1 - v_3$	4.36
v_4	$v_0 - v_1 - v_3 - v_4$	6.86
v_5	$v_0 - v_1 - v_3 - v_4 - v_5$	9.16
v_6	$v_0 - v_1 - v_3 - v_4 - v_5 - v_6$	10.46
v_7	$v_0 - v_1 - v_3 - v_4 - v_5 - v_6 - v_7$	11.96
v_8	$v_0 - v_1 - v_3 - v_4 - v_5 - v_6 - v_7 - v_8$	13.66
v_9	$v_0 - v_1 - v_3 - v_4 - v_5 - v_6 - v_7 - v_8 - v_9$	15.26
v_{10}	$v_0 - v_1 - v_3 - v_4 - v_5 - v_6 - v_7 - v_8 - v_9 - v_{10}$	16.76
v_{11}	$v_0 - v_1 - v_3 - v_4 - v_5 - v_6 - v_7 - v_8 - v_9 - v_{10} - v_{11}$	18.36
v_{12}	$v_0 - v_1 - v_3 - v_4 - v_5 - v_6 - v_7 - v_8 - v_9 - v_{10} - v_{11} - v_{12}$	19.86
v_{13}	$v_0 - v_1 - v_3 - v_4 - v_5 - v_6 - v_7 - v_8 - v_9 - v_{10} - v_{11} - v_{12} - v_{13}$	21.06
v_{14}	$v_0 - v_1 - v_3 - v_4 - v_5 - v_6 - v_7 - v_8 - v_9 - v_{10} - v_{11} - v_{12} - v_{13} - v_{14}$	22.96
v_{15}	$v_0 - v_1 - v_3 - v_4 - v_5 - v_6 - v_7 - v_8 - v_9 - v_{10} - v_{11} - v_{12} - v_{13} - v_{14} - v_{15}$	23.76

The proposed modified incident edges path algorithm solves the pipeline network problem and is verified using MATLAB. For example, the following Figure 2 gives the required optimum path for the network. The red line in Figure 2 denotes the optimum path to reach the final pump station from the origin with minimum transportation cost.

II. Google Map Network

The Google Map Network Problem is taken as a numerical example to find the optimum path with minimum transportation cost using the proposed MIEPA algorithm. A map is considered for finding the optimum route between Avadi and Ambattur is shown in Figure 3. The Google map shows two types of timings, such as 20 minutes with 8.8km and 24 minutes with 9.4km to reach the destination as the fastest route. But using the proposed MIEPA algorithm, the decision-maker can find the shortest route with a minimum time duration compared with the online server, which was solved and provided in below Table 5 and Figure 5.

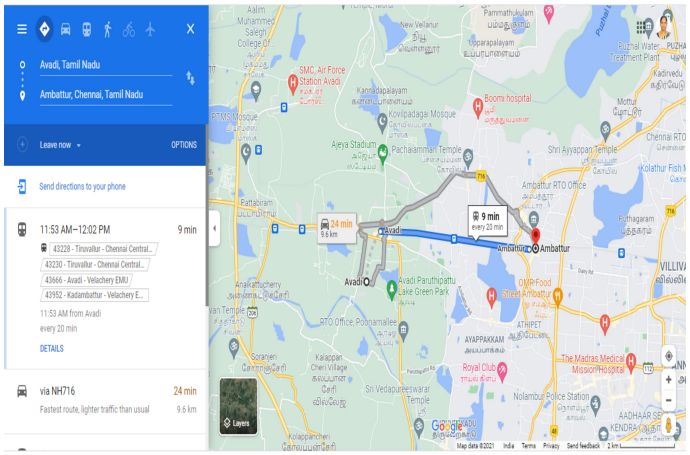

Figure 3: *Route taken from online server - Google Maps*

The distance may vary because of the parameters like speeding limits and road extension work. The distance taken for calculating is taken from the Google Server and solved by the proposed MIEPA algorithm. Figure 4 shows the graphical representation of the google map; each intersection area is taken as nodes, the line joining nodes is taken as edges, and the time taken to reach between nodes is considered as distances of edges. The network taken is directed to reach the destination using Google Maps and plotted using MATLAB, which is a directed network. The nodes are

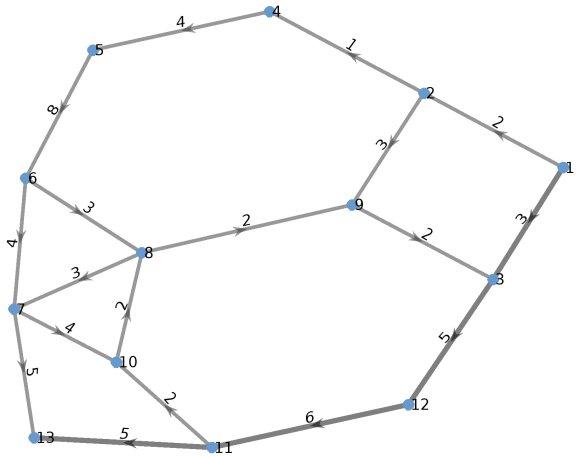

Figure 4: *Network of Google Map*

labeled as $\{1, 2, 3, ..., 12, 13\}$ representing the nodes $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}\}$ where v_1 denotes Avadi, v_{13} denote's Ambattur and distance between nodes is given in Figure 4. The Table 5 provides the distances between nodes for the Google Map Network Problem.

Table 5: *Edge and Distance Between the Nodes of Google Map Network*

Edge	**Distance Between v_i and v_j**	**Edge**	**Distance Between v_i and v_j**
(v_1, v_2)	2	(v_7, v_{10})	4
(v_1, v_3)	3	(v_7, v_{13})	5
(v_2, v_9)	3	(v_8, v_7)	3
(v_2, v_4)	1	(v_8, v_9)	2
(v_3, v_{12})	5	(v_{11}, v_{13})	5
(v_4, v_5)	4	(v_9, v_3)	2
(v_5, v_6)	8	(v_{10}, v_8)	2
(v_6, v_7)	4	(v_{12}, v_{11})	6
(v_6, v_8)	3	(v_{11}, v_{10})	2

Table 6 provides the calculation process by proposed MIEPA for Google Map Network Problem for optimum route between Avadi to Ambattur.

Table 6: *Google Map Network Problem Solved by MIEPA (Part I)*

Incident Edge Set E_j	**The Weight and the Shortest Path p_j of (v_j)**
$E_1 = \{v_1v_3, v_1v_2\}$	$p_3 = 3$ of (v_3) and $p_2 = 2$ of (v_2)
$E_2 = \{v_2v_4, v_2v_9\}$	$p_4 = 3$ of (v_4) and $p_9 = 5$ of (v_9)
$E_3 = \{v_4v_5\}$	$p_5 = 7$ of (v_5)
$E_4 = \{v_5v_6\}$	$p_6 = 15$ of (v_6)
$E_5 = \{v_6v_7, v_6v_8\}$	$p_7 = 19$ of (v_7) and $p_8 = 18$ of (v_8)
$E_6 = \{v_8v_7, v_8v_9\}$	$p_7 = \min\{19, 21\} = 19$ of (v_7) and $p_9 = \min\{5, 20\} = 5$ of (v_9)
$E_7 = \{v_9v_3\}$	$p_3 = \min\{3, 7\} = 3$ of (v_3)
$E_8 = \{v_3v_{12}\}$	$p_{12} = 8$ of (v_{12})
$E_9 = \{v_{12}v_{11}\}$	$p_{11} = 14$ of (v_{11})
$E_{10} = \{v_{11}v_{10}, v_{11}v_{13}\}$	$p_{10} = 16$ of (v_{10}) and $p_{13} = 19$ of (v_{13})
$E_{11} = \{v_{10}v_8\}$	$p_8 = \min\{16, 18\} = 16$ of (v_8)
$E_{12} = \{v_8v_9\}$	This forms a cycle. Neglect this path and choose the path (v_{11}, v_{13}) leading to end node v_{13}.
$E_{13} = \{\}$	$p_{13} = 19$

End node reached, so stop the process since $E_{13} = \{\}$ there is no edges incident with v_{13} and weight of the end node is selected which is the minimum weight of path. Hence, the optimum time to reach the destination is 19 minutes

Table 7: *Shortest Path for Each Node from v_1 in the Google Map Network*

End Node	**Shortest Path $p(v_1, v_j)$**	**Minimum Cost w_j of $p(v_1, v_j)$**
v_2	$v_1 - v_2$	2
v_3	$v_1 - v_3$	3
v_4	$v_1 - v_2 - v_4$	3
v_{12}	$v_1 - v_3 - v_{12}$	8
v_9	$v_1 - v_2 - v_9$	5
v_{11}	$v_1 - v_3 - v_{12} - v_{11}$	14
v_5	$v_1 - v_2 - v_4 - v_5$	7
v_6	$v_1 - v_2 - v_4 - v_5 - v_6$	15
v_8	$v_1 - v_2 - v_4 - v_5 - v_6 - v_8$	18
v_7	$v_1 - v_2 - v_4 - v_5 - v_6 - v_7$	19
v_{10}	$v_1 - v_3 - v_{12} - v_{11} - v_{10}$	16
v_{13}	$v_1 - v_3 - v_{12} - v_{11} - v_{13}$	19

The Table 7 provides the optimum path with minimum time to reach the different destination from the origin node v_1 by visiting all the edges. Hence, the destination reached from origin and optimum path is $v_1 - v_3 - v_{12} - v_{11} - v_{13}$ with minimum time duration (Weight of the path) of $p_{13} = 19$ minutes.

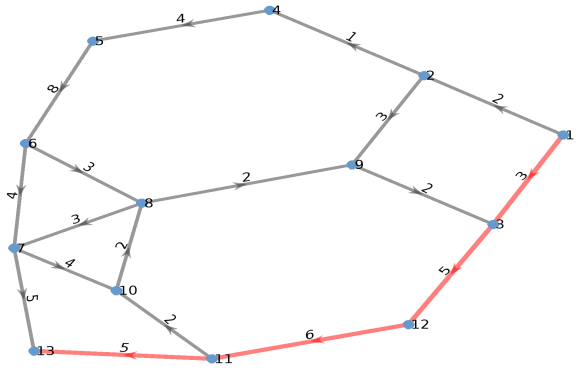

Figure 5: *optimum path of Google Map by MATLAB*

The Figure 5, provide the optimum path for a specified map taken from Google Maps found using the proposed MIEPA algorithm in MATLAB Hence, the decision-maker can find the optimum path by taking any node as the origin and any node as the destination. Therefore, the proposed MIEPA algorithm provides the minimum transportation cost, optimum path, and minimum time duration.

IV. Discussion and Conclusion

The classical network optimization problem of finding the optimal path is crucial in numerous real-world applications, particularly in smart logistics. This article implements the Modified Incident Edge Path Algorithm (MIEPA) to address smart logistical challenges in two real-life scenarios. The MIEPA algorithm efficiently determines the optimal path with minimum transportation cost and time duration by utilizing incident edges to traverse the network.

The algorithm's running time complexity is reduced to $O(E + V)$, offering an efficient solution compared to existing algorithms. The paper presents a comparison of MIEPA with other algorithms, highlighting its performance advantages. Additionally, the algorithm ensures that the destination node is reached from the origin by visiting all edges in a given directed network. The optimal path identified by MIEPA for both pipeline network and Google Map problems is verified using MATLAB, achieving minimum transportation costs. The algorithm also provides results consistent with those obtained from Dijkstra's algorithm, as shown in [3] and [9]. MIEPA demonstrates superior performance in determining the shortest and fastest routes compared to Google Search for the Google Map Problem. Furthermore, the paper illustrates how MIEPA can calculate multiple optimal paths between various origin and destination pairs, enhancing decision-making for users. In conclusion, the MIEPA algorithm facilitates easy manual determination of the shortest route and provides a straightforward, efficient solution for reaching all nodes from the origin node or between any specified nodes in sparse directed network problems. Its main advantage is its clarity and efficiency in solving smaller network instances.

References

[1] O. T. Arogundade and A.T. Akinwale A. T., (2009), Application of Prim™s Algorithm to a Profit- Oriented Transportation System in Rural Areas of Nigeria, University of Agriculture, Abeokuta, 37, pp.4-9.

[2] Avdhesh, K.S. and Sourabh, K., (2013), Finding of Shortest Path from Source to Destination by traversing every Node in a wired Network, International Journal of Engineering and Technology, 5, pp.2655-2656.

[3] Chu Fei-xue and Chen Shi-yi, (2012), Optimal Design of Pipeline based on the Shortest path, International Conference on Medical Physics and Biomedical Engineering, Physics Procedia., 33, pp.216-220.

[4] Deo, Narsingh, (1974), Graph Theory with Applications to Engineering and Computer Science, Englewood, New Jersey: Prentice-Hall.

[5] Douglas B. West, (2009), Introduction to Graph Theory, PHI NewDElhi, ED-2.

[6] Hamdy A. Taha, (2013), Operations Research: An Introduction , 9th ed. 978 0132555937

[7] Kairanbay, M., Hajar. M.J., (2013), A Review and Evaluations of Shortest Path Algorithms, International Journal of Scientific and Technology Research, 2, pp.99-101E.

[8] Rishi Pal Sing and Vandana, (2014), Application of Graph Theory in Computer Science and Engineering, International Journal of Computer Applications, 104(1), pp.0975 " 8887.

[9] Sathyapriya S, Kavin M K and Mythreye Rakshana S, (2020), Implementation of Dijkstras Algorithm to Find the Shortest Path in Google maps, International Journal of Creative Research thoughts(IJCRT), pp.2320-2882.

[10] Pandian P and Rajendran P, (2010), A new algorithm for minimum path in as network, Applied Mathematical Sciences, 4(54), pp.2697-2710.

[11] Xia Song and Han Yong-Shu, (2004), An Improved Implementation Of Shortest Path Algorithm In GIS, Bulletin Of Surveying And Mapping, 9, pp.40-42.

[12] Yue Yang and Gong Jian-Ya, (1999), An Efficient Implementation Of Shortest Path Algorithm Based On Dijkstra Algorithm, Journal Of Wuhan Technical University Of Surveying And Mapping, 24(4), pp.209-212.

[13] Zrinka L. , Dubravko H., and Luka N, (2008), Solving The Production-Transportation Problem In The Petroleum Industry, Revista Investigacin Operacional, 29(1), pp.63-70.

[14] Yassine Issaoui, (2022), An Advanced System to Enhance and Optimize Delivey Operations in a Smart Logistics Environment, IEEE Access, 10, pp.6175-6193.

[15] Jiajun Hu, (2021), The Shortest Path Application of the Logistics System in Production Workshop, 4th International Conference on Data Science and Information Technology, 978-1-4503-9024-8.

[16] Yogesh chanchal, (2015), Analysis The Time Complexity of Greedy Algorithm Used in Optimization, IJCAT - International Journal of Computing and Technology, 2(7), pp.2348 - 6090.

[17] Kanchana M and Kavitha K, (2023), A Comparative Study Between Incident Edge Path Algorithm and Dijkstra Algorithm for Finding Shortest Path for Networks, "Novel Developments in Computational Intelligence systems and their Applications in Multidisciplinary areas", book chapter, "https://novapublishers.com/product-category/series/computer-science-.".