Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 07, July-2024

# Finding the Shortest Path in the Transport Problem using Dijikstra's Algorithm

Vijayalaxmi M K
MGM University, Aurangabad

Dr.Tanaji Pawar
MGM University, Aurangabad

***Abstract*: Dijkstra's Algorithm is a fundamental method for finding the shortest path in weighted graphs, with applications ranging from network routing to logistics and supply chain management. This algorithm begins by initializing distances from a source node to all others, then iteratively selects the node with the shortest known distance, updating distances to its neighbour's. The process continues until all nodes are visited, determining optimal paths. In logistics, it helps optimize transportation routes, balancing factors like cost and time. While Dijkstra's Algorithm is efficient and versatile, research gaps exist, such as scalability for large networks, real-time adaptation, and handling uncertainty. Nonetheless, its efficiency, flexibility, and practical use cases underscore its significance in addressing transportation and logistics challenges. Dijkstra's Algorithm, originally designed for finding the shortest path in a graph, can be adapted to solve the Transport Problem efficiently. The algorithm starts at an initial node (supply point), explores adjacent nodes (routes) to calculate the cumulative cost of reaching them, and selects the path with the minimum cost. This process continues iteratively, updating cost estimates until the algorithm reaches the destination (demand point) or exhausts all possible routes.**

***Keywords:* Transport Problem, Dijkstra's Algorithm, Optimization, Logistics, Supply Chain Management**

## I. INTRODUCTION:

This paper discusses the development of an efficient method for determining the best travel route between two points, resulting in the creation of the shortest path algorithm. Known as Dijkstra's Algorithm, this graph search technique is designed to solve the single-source shortest path problem within a graph, specifically one with non-negative edge path costs. Its primary application lies in network routing and related protocols[1].
Over time, Dijkstra's Algorithm has demonstrated its versatility, extending its utility beyond its initial purpose in network routing to various domains, including logistics and supply chain management. Within these contexts, the algorithm plays a crucial role in optimizing transportation routes, taking into account factors like cost and time while addressing supply and demand constraints. This paper explores the adaptation of Dijkstra's Algorithm to efficiently solve transportation problems, with a focus on optimizing transportation processes and using mathematical models to represent complex logistics scenarios. Additionally, it delves into multi-criteria decision-making techniques, time-dependent analysis, and practical applications of the algorithm in real-world logistics challenges[2]. Leveraging the algorithm's efficiency, flexibility, and adaptability can enhance organizations' logistics operations, ultimately contributing to improved efficiency and cost-effectiveness in transportation and supply chain management[3].
For a given source vertex or node in the graph, the algorithm computes the shortest path to a single destination vertex. Once the shortest path to the destination vertex is determined, the algorithm terminates. To illustrate, if the graph's vertices represent cities and edge path costs represent driving distances between city pairs connected by direct roads, Dijkstra's algorithm can find the shortest route from one city to all others. Extensive research has been conducted on finding shortest paths within such networks[4].
Dijkstra's algorithm is specifically designed for finding the shortest path in a directed graph with edges weighted by non-negative values. [5]. Dijkstra's algorithm is recognized as asymptotically the fastest known single-source shortest-path algorithm for arbitrary directed graphs with unbounded nonnegative weights. These motivations drive our quest to further understand Dijkstra's algorithm in the future[6].

## II. PURPOSE METHODOLOGY:

- Algorithm Adaptation: Demonstrate how Dijkstra's Algorithm can efficiently solve transportation problems by representing suppliers and consumers as nodes with transportation costs as edge weights.
- Optimization: Emphasize the goal of optimizing transportation processes, including cost and time minimization, while meeting supply and demand constraints.
- Mathematical Modelling: Describe the mathematical modelling used to represent transportation problems, including objective functions, decision variables, and supply and demand constraints.
- Multi-Criteria Decision-Making: Discuss the application of MCDM techniques to optimize transportation routes based on multiple criteria and the calculation of combined scores.
- Time-Dependent Analysis: Introduce time-dependent analysis in transportation problems and how time-dependent cost functions account for dynamic factors like traffic and time variations.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 07, July-2024

## III. DIJKSTRA'S ALGORITHM CONCEPT:

Dijkstra's Algorithm is a well-known algorithm in computer science and mathematics for determining the shortest path between two nodes in a weighted network. Edsger.W.Dijkstra created it in 1956 and it is particularly beneficial for tackling different network routing and optimization challenges.[7].

Here's an overview of how Dijkstra's Algorithm works and its techniques:

1. Initialization: The algorithm starts by initializing a list to keep track of the shortest distance from a selected source node to every other node in the network. At first, all distances are set to infinity except for the source node itself, which is set to a distance of 0.
2. Exploration Queue: Dijkstra's Algorithm to keep track of the nodes to be investigated, a priority queue (typically implemented as a min-heap) is used. The priority queue is initially assigned to the source node.
3. Main Loop: The algorithm enters a loop that continues until the priority queue is empty or until a specific target node (if provided) is reached[8].

- Extracting the Minimum: In each iteration of the loop, the algorithm extracts the node with the minimum distance from the priority queue. This node is the one with the currently shortest known distance from the source.
- Relaxation: The approach estimates a tentative distance from the source node through the extracted node for each nearby node of the extracted node. If this tentative distance is less than the neighbour's current known distance, the distance is updated.
- Adding to the Queue: If the distance to a neighbour node is updated, or if the neighbour hasn't been visited yet, it is added to the priority queue for further exploration[9].

4. Termination: When all nodes have been visited or a specified target node has been reached, the algorithm finishes. The algorithm has now determined the shortest path from the source node to every other node in the network.

- Initialize a list to keep track of the shortest distance between the source node and each node in the graph. All distances are set to infinity except for the source node, which is set to 0.Initialize a set $S$ to keep track of visited nodes. Initialize a list $d$ to keep track of the shortest distance from the source node to each node in the graph. Initialize all distances to infinity except for the source node: $d[s] = 0$[10].
- Create a priority queue (min-heap) $Q$ to keep track of the nodes to be explored. Initially, add the source node to $Q$[11].While the priority queue Q is not empty:

○ Extract the node u with the minimum distance from Q:
○ *u=extractMin (Q)*

- For each neighbour of the extracted node:
- Calculate the tentative distance from the source node to the neighbour through the extracted node.

○ *Tentative distance*=$d[u] + w(u, v)$

- If this tentative distance is less than the current distance recorded for the neighbour, update the neighbour's distance.

○ $d[v]$=*tentative distance*
○ Add the neighbour to the priority queue if it hasn't been visited yet.

Once all nodes have been visited or the target node is reached, the algorithm terminates. The shortest distance to each node from the source node is now known[12].

An Example of Dijkstra's Algorithm:
The following is the procedure for implementing Dijkstra's Algorithm:
Step 1: Initially, set the distance of the source node to 0, and assign a distance of INFINITY to all other nodes.
Step 2: Identify the unvisited node with the shortest current distance, denoted as X, and designate it as the current node.
Step 3: For each neighbour N of the current node X, calculate a tentative distance by adding the current distance of X to the weight of the edge connecting X to N. If this tentative distance is shorter than the current distance of N, update N's distance to this new value.
Step 4: Mark the current node X as visited.
Step 5: If there are still unvisited nodes in the graph, return to 'Step 2' and repeat the process.
Let us now look at how the algorithm is implemented using an example:

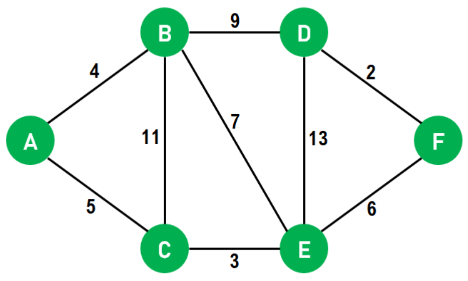

Figure 1: Algorithm of Time Complexity

1. Start with a graph and designate a source node (in this case, node A).
2. Initialize all nodes as unvisited.
3. Set the path to the source node (A) as 0 and all other nodes' paths as infinity.
4. Mark the source node A as visited (or accessed).
5. Use relaxation to update the path to neighbouring nodes (in this case, nodes B and C) based on the path to node A. Update the path to the node with the minimum distance among the unvisited neighbors.
6. Continue relaxing nodes until all of the source node's neighbours have been visited.
7. Move to the next unvisited node with the shortest path (in this case, node B) and relax its unvisited neighbours (nodes C, D, and E).
8. Repeat the relaxation process for other unvisited nodes until all nodes are visited. This ensures that you find the shortest path through the network from the source node to all other nodes
9. Once all nodes have been visited, the algorithm ends, and you have the shortest routes between the source node and every other node in the network.

As a result, final pathways we arrived at are:
A = 0
$B = (A \geq B)$
$C = 5(A \geq C)$
$D = 4 + 9 = 13(A \geq B \geq D)$
$E = 5 + 3 = 8(A \geq C \geq E)$
$F = 5 + 3 + 6 = 14(A \geq C \geq E \geq F)$
A is assigned a value of 0. This is straightforward and doesn't rely on any conditions is assigned a value of 0 because it depends on the condition $A \geq B$. In this case, since A is 0 and 0 is indeed greater than or equal to 0, the condition is met, resulting in B being assigned the value of 0.C is assigned a value of 5 because it depends on the condition $A \geq C$. Since A is 0, and 0 is greater than or equal to 5, this condition holds, leading to C being assigned the value of 5.D is assigned a value of 13 because it relies on two conditions. First, $A \geq B$, which is true as explained earlier, and second, $B \geq D$. Given that B is also 0, it is indeed greater than or equal to D (which is 13 in this context), resulting in D being assigned the value of 13.E is assigned a value of 8 because it depends on the conditions $A \geq C$ and $C \geq E$. As established earlier, both of these conditions are true (A is greater than or equal to C, and C is greater than or equal to E), so E is assigned the value of 8.F is assigned a value of 14, which depends on three conditions: $A \geq C$, $C \geq E$, and $E \geq F$. All of these conditions are true based on the previously established values (A is greater than or equal to C, C is greater than or equal to E, and E is greater than or equal to F). Therefore, F is assigned the value of 14.

## IV. LITERATURE SURVEY:

Dijkstra's Algorithm, originally designed for finding the shortest path in networks, has been adapted to address the Transport Problem. This literature review explores various applications of Dijkstra's Algorithm in solving the Transport Problem, examining the mathematical models and equations employed.

1. Theoretical Foundation:
The Dijkstra Algorithm is a well-known graph traversal approach for determining the shortest path between nodes in a weighted graph. In the context of the Transport Problem, it can be applied by representing suppliers and consumers as nodes and transportation costs as edge weights.
2. Mathematical Model and Equation:
The mathematical model for the Transport Problem using Dijkstra's Algorithm can be expressed as follows:

Sets:
- S: Set of suppliers (i.e., sources of goods).
- C: Set of consumers (i.e., destinations for goods).

Parameters:
- d_ij: Cost of transporting one unit of goods from supplier i to consumer j.
- x_ij: Decision variable representing the goods transportable supplier I the consumer j[13].

Objective Function:
- Minimize total transportation cost[14]:

$$Z = \sum_{i \in S} \sum_{j \in C} dij.xij$$

Constraints:

- Supply Constraint:

$$\sum_{j \in C} xij \leq supply\ i, for\ all\ i \in S$$

- Demand Constraint:

$$\sum_{i \in S} xij \geq Demandj, for\ all\ j \in C$$

- Non-negativity Constraint:

$$xij \ \geq 0, for\ all\ i \ \in S, j \ \in C$$

In this equation:

- Z represents the total transportation cost, which we aim to minimize[15].
- i and j are indices representing suppliers and consumers, respectively.
- $d_{ij}$ represents the cost of transporting one unit of goods from supplier *i* to consumer *j*.
- $X_{ij}$ represents the decision variable that indicates the quantity of products to be carried from supplier i to consumer j.
- Then constraints ensure that supply and demand requirements are met and that the transportation quantities are non-negative[16].

Dijkstra's Algorithm could be used within this framework to help determine the shortest paths and transportation costs ($d_{ij}$) between suppliers and consumers, which are then used in the linear programming model to optimize the logistics. However, the optimization problem itself is typically expressed in the linear programming or integer programming format, as shown above[17].

## 3. Literature Table:

| Title | Year | Techniques | Mathematical Model | Description |
|---|---|---|---|---|
| 1.Hazardous material transportation problems: A comprehensive overview of models and solution approaches[18]. | 2021 | Various modelling and solution techniques | Transportation Problem (Linear Programming): $Z = \sum_{i=1}^{m} \sum_{j=1}^{n} CijXij$ Supply Constraints: $\sum_{j=1}^{n} xij \leq ai\ for\ i = 1,2,..,m$ Demand constraints: $\sum_{i=1}^{m} xij \geq bj\ for\ j = \ 1,2,..,n$ | The transportation problem is a type of linear programming problem that deals with finding the most cost-effective way to transport goods from multiple suppliers to multiple consumers while satisfying supply and demand constraints.$c_{ij}$ represents the cost of shipping from supplier *i* to consumer *j*, and $x_{ij}$ represents the quantity to be shipped. |
| 2.Minimization of the probability of serious road accidents in the transport of dangerous goods[19]. | 2022 | Route determination, statistical analysis | $Z = f(R) + g(S)$ Where: • *Z* represents the objective function to be minimized. • *f*(*R*) represents the route determination component, which considers factors such as distance, time, or cost for the selected route. • *g*(*S*) represents the statistical analysis component, which accounts for safety or risk-related factors associated with the chosen route[20]. | In this equation, the objective is to find a route *R* that minimizes the combined cost *f*(*R*) and maximizes safety *g*(*S*), taking into account the specified weights. This approach enables the careful equilibrium between optimizing transportation routes for efficiency while also considering safety in the planning process. Please note that the specific forms of *f*(*R*) and *g*(*S*) would depend on the details of the transportation problem and the metrics or measures used to quantify cost and safety. Additionally, optimization techniques such as linear programming or heuristic algorithms may be used to solve this objective function in practical applications. |
| 3.A new time-dependent shortest path algorithm for multimodal transportation network[21]. | 2017 | Time-dependent shortest path algorithm, heuristics | C(v,t)=Base Travel Time(v)+Time-Varying Component(v,t) $*C(v,t) = v + t$ • Base Travel Time (v): The base travel time or static component representing the travel time in the absence of dynamic factors. • Time-Varying Component (v, t): The time-dependent component that | The time-dependent cost function, denoted as C (v, t), represents the time-dependent travel cost or time required to traverse a specific edge or segment. It takes into account factors like traffic congestion, time-of-day variations, and other dynamic conditions. |

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 07, July-2024

| | | | | |
|---|---|---|---|---|
| | | | captures variations in travel time due to factors like congestion, traffic signals, or speed fluctuations. | |
| 4.Implementation of Dijkstra Algorithm and Multi-Criteria Decision-Making for Optimal Route Distribution[16]. | 2019 | Dijkstra Algorithm, Multi-Criteria Decision-Making | Combined Score: $C = wD.D + wT.T$ In this equation: <br>• $C$ represents the combined score. <br>• $D$ represents the distance of the route. <br>• $T$ represents the travel time of the route. <br>• $wD$ and $wT$ are the weights assigned to distance and time, respectively. <br>• Distance (D): Represents the physical distance of the route. <br>• Time (T): Represents the travel time of the route. | Dijkstra's algorithm to find the shortest path based on one criterion (e.g., distance) and then calculate the combined score for each route using the weighted sum approach. The methods used in Multi-Criteria Decision-Making can vary depending on the technique chosen and the details of the problem. The example provided here demonstrates a basic approach for combining Dijkstra's algorithm with multiple criteria. |
| 5.From the Digital Internet to the Physical Internet: A Conceptual Framework With a Stylized Network Model[22]. | 2021 | Conceptual Framework Graphics Analysis | Flow Conservation Equation: $\sum_j xij - \sum_j xji = 0$ In this equation: <br>• $xij$ represents the flow from node $i$ to node $j$. <br>• The first summation $\sum j.xij$calculates the total flow leaving node $i$. <br>• The second summation $\sum j.xji$ calculates the total flow entering node $i$. | In transportation and network flow problems, one of the fundamental principles is flow conservation. This principle states that, at any intermediate node in a transportation network, the total flow into the node equals the total flow out of the node. The equation essentially states that the net flow into or out of nod i must be zero, ensuring that flow is conserved at every node in the transportation network. Please note that this is a simple example to illustrate a concept within transportation modelling. More complex transportation models and frameworks may involve a series of equations and constraints to represent various aspects of transportation network design, optimization, or flow. |
| 6.On the Robust Shortest Path Problem[23]. | 2017 | Robust shortest path problem, scenario approach | Objective Function (Minimize Worst-Case Cost): $Minimize\ max_s \sum_{e \in E} Ce,s.Xe$ In This equation: <br>• Maxs represents the maximum over all scenarios. <br>• This equation calculates the total cost of the selected path for a specific scenario s <br>• Xe is a binary decision variable representing whether edge e is included in the path <br>• Ce,s represents the cost of edge e under scenarios s. | This objective function aims to minimize the maximum (worst-case) cost among all scenarios, ensuring that the selected path performs well even under the most unfavourable scenario. This formula encapsulates the essence of the Robust Shortest Path Problem using a scenario-based approach, where the goal is to find a path that is robust to uncertainty in edge costs by minimizing the worst-case cost over a set of scenarios. |
| 7.From the Digital Internet to the Physical Internet: A Conceptual Framework With a Stylized Network Model with Dijkstra Algorithm [24]. | 2021 | Conceptual framework, graph theory-based model | $* \ TF = Supply - Demand$ In this equation: <br>• TF means "Transport Flow" represents the flow of goods or resources in the transportation network. <br>• "Supply" represents the total supply available at the source or origin nodes. <br>• "Demand" represents the total demand at the destination nodes. | This equation captures the fundamental concept of flow balance in transportation networks, indicating that the flow of goods or resources from sources to destinations must balance supply and demand. It's a simplified representation of a transportation model that adheres to flow conservation principles and can serve as a foundational equation in transportation network analysis. |

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 07, July-2024

| | | | | |
|---|---|---|---|---|
| 8.Graph representation learning: a survey[25] . | 2020 | Graph embedding techniques, evaluation | Given two sets of node embedding's:<br>• Embedding's for graph A: $X = \{X_1, X_2, \dots\dots, Xn\}$, where *Xi* is the embedding for node *i* in graph A<br>• Embedding's for graph B: $Y = \{Y_{1,} Y_{2,\dots\dots,} Ym\}$<br>Where *Yj* is the embedding for node *j* in graph B. | The goal is to find an optimal transport plan, often represented as a matrix *P*, where *Pij* represents the quantity of "mass" transferred from node i in graph A to node j in graph B while minimizing the overall transportation cost. Then the transport equation seeks to minimize the cost or discrepancy between the two embedding's under the constraint that the total mass transported equals 1 for each node and that the transport plan *P* is non-negative. The cost of transporting mass from *xi* to *yj* is often defined as a distance metric, such as the Euclidean distance or the squared Euclidean distance: |
| 9. How Much Will the Belt and Road Initiative Reduce Trade Costs[26]. | 2018 | Geographic Information System (GIS) analysis, estimation of ad valorem trade costs, sectoral analysis | Trade Cost Equation:<br>$$TC_{ij} = \frac{D_{ij.VOT}}{S_{ij}}$$<br>Where:<br>• *TCij* represents the ad valorem trade cost from location *i* to location *j*.<br>• *Dij* is the distance or shipment time between location *i* and location *j* as obtained from GIS analysis.<br>• *VOT* is the "Value of Time," which represents the monetary value assigned to the time savings for the shipment.<br>• *Sij* is a sector-specific factor that can represent various sectorial considerations, such as transportation mode-specific costs, tariffs, or other sector-specific factors that influence trade costs. | This equation simplifies the estimation of ad valorem trade costs by taking into account the distance or shipment time between locations, the sector-specific factors, and the value assigned to time savings (Value of Time).<br>Please note that this is a simplified representation, and in practice, trade cost estimation can be much more complex, considering a wide range of factors such as infrastructure quality, border delays, administrative costs, and more. The specific equation and factors used for trade cost estimation can vary depending on the research context and available data. |
| 10. Approximation Schemes for the restricted shortest path problem[27]. | 2017 | shortest path problem, polynomial-time algorithms | Constrained Shortest Path Equation:<br>$$f(x) = \sum_{i,j} C_{ij} X_{ij}$$<br>• *Xij* ∈ {0,1} For all edges (*i,j*) in the graph, where *xij* is a binary decision variable that indicates if an edge exists (*i,j*) is chosen in the path.<br>• $\sum_{i,j} a_{ij} X_{ij} \leq B$ Where *aij*represents a constraint associated with edge (*i,j*), and *B* is the maximum allowable constraint value. | This is a basic formulation of a constrained shortest path problem, where you aim to find the path through a graph with binary decision variables<br>Xij while satisfying a constraint related to the selected edges. To approximate this problem with a polynomial-time algorithm, you might employ techniques like linear programming relaxation and rounding. The relaxation typically involves relaxing the binary constraint to continuous values between 0 and 1, which results in a linear programming problem that can be solved efficiently. Afterward, a rounding scheme can be applied to obtain an approximate binary solution. Keep in mind that the specific constraints, costs (cij), and constraint values (B) would depend on the problem context, and the approximation algorithm used would be chosen based on the problem's complexity and requirements. |
| 11.Shortest path problem on uncertain networks: An efficient two<br>phases approach[28]. | 2021 | Networks, graph modeling, two-phase approach, uncertainty handling. | Uncertainty-Aware Network Flow Equation:<br>$$Maximize\ Z = \sum_{i,j} (x_{ij} - y_{ij}).P_{ij}$$<br>• $x_{ij} \leq U_{ij}$For all edges (*i,j*), where *xij* represents the flow on edge (*i,j*), and *Uij* is the maximum capacity of that edge.<br>• $x_{ij} \geq 0$ For all edges (*i,j*). | This equation represents an optimization problem where you maximize a weighted sum of the difference between the actual flow (xij) and the uncertain flow (yij) on network edges. The uncertainty is modelled using an upper bound (Uij), which can represent factors like traffic congestion. |

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 07, July-2024

| | | | | |
|---|---|---|---|---|
| | | | • $\sum_i xij - \sum_k xki$ for all nodes *j*, where *dj* is the demand or supply at node *j*.<br>• $y_{ij} \leq U_{ij}$For all edges (*i*,*j*), where *yij* represents the uncertainty (e.g., due to traffic conditions) on edge (*i*,*j*), and *Uij* is the upper bound on that uncertainty.<br>• $y_{ij} \geq 0$ For all edges (*i*,*j*). | The two-phase approach would involve:<br>Phase 1 (Pre-processing): In this phase, you might estimate or model the uncertainty (yij) on the network edges based on available data or predictive models. This could involve gathering real-time traffic information, weather conditions, or other relevant factors.<br>Phase 2 (Query or Optimization): Once you have estimates for the uncertainties, you can use the equations above to optimize the network flow (xij) while considering these uncertainties, ensuring that the flow remains within the capacity limits (Uij) and satisfies demand constraints. This is a simplified example, and in practice, uncertainty modeling and handling in network optimization can be much more complex, involving probabilistic or stochastic models, scenario analysis, and more advanced techniques. The specific equations and approaches used would depend on the problem context and the nature of the uncertainties involved. |
| 12. SHORTEST PATH METHODS: A UNIFYING APPROACH[29]. | 2015 | Shortest path algorithms, data structures for shortest path. | Uncertain Shortest Path Equation:<br>$G = (V, E)$<br>• Where *V* represents nodes (locations) and *E* represents edges (connections between nodes). Each edge *e*∈*E* has an associated uncertain weight *we*, representing the uncertain travel time or cost along that edge.<br>• **Graph Modelling:** In the graph, let *i* and *j* represent two nodes connected by an edge *eij*, and *xij* be a binary decision variable indicating whether edge *eij* is selected in the path.<br>• Optimization $minimize =$ $E[C]$=$\sum_{eij \subset E} xij. E[Weij]$ | **Phase 1 (Pre-processing):** In this phase, you estimate the probability distribution of the uncertain edge weights *We* for all edges in the graph. For example, you might represent *We* as random variables following specific probability distributions.<br>**Phase 2 (Optimization):**<br>Once you have estimates of the uncertain edge weights, you can formulate the uncertain shortest path problem as follows:<br>where $E[C]$ is the expected cost (or expected travel time) of the path, $E[Weij]$ is the expected weight of edge *eij* based on the probability distribution of *Weij*, and *xij* is a binary decision variable indicating whether edge *eij* is part of the path. |
| 13.Stochastic Shortest Path: Minimax, Parameter-Free and Towards Horizon-Free Regret[30]. | 2021 | Stochastic shortest path, minimax regret, exploration bonuses, parameter-free algorithms | The Minimax Regret *R* over a horizon-free setting can be represented as<br>$R = max_s(J_{T(s)} - min_a J_T(s,a))$<br>• R represent the Minimax Regret represent the time horizons represent a state or node a represent an action (path choice).<br>• JT(s,a) represent the expected cost-to-go for state s when taking action a over the first T stages.<br>• JT(s) represent the minimum expected cost-to-go for state's over the first T stages, considering all possible actions. | This equation captures the idea of minimizing the maximum regret over all states regardless of the time horizon, where<br>JT(s) represents the minimum expected cost-to-go for states over an unspecified time horizon, and min(s,a) represents the best expected cost-to-go achieved by taking any action over the same unspecified time horizon.<br>In practice, the computation of R may involve dynamic programming, reinforcement learning algorithms, or other optimization techniques to find the optimal policy that minimizes this regret criterion while considering exploration bonuses and adapting to the environment without predefined parameters (parameter-free algorithms). |
| 14.Application of Dijkstra Algorithm in Logistics Distribution Lines[31]. | 2015 | Dijkstra Algorithm for path optimization in | Tentative distance alt can be written as:<br>$alt = d(u) + w(u, v)$ | This equation represents the sum of the current shortest distance to node *d*(*u*)) and the weight of the edge from *u* to its neighbor (*w*(*u*,*v*)). If *alt* is |

| | | | | |
|---|---|---|---|---|
| | | logistics distribution lines | | smaller than the current shortest distance to *v* $d(v)$), it means that there is a shorter path from the source to *v* through *u*, so $d(v)$ is updated with this shorter distance. This process continues until the shortest paths to all reachable nodes have been found, providing an optimized path in logistics distribution lines based on the chosen weight (distance, time, or cost) criteria. |
| 15.Surface Optimal Path Planning Using an Extended Dijkstra Algorithm[32]. | 2020 | extended Dijkstra Algorithm for surface path planning | Extended Dijkstra Algorithm $w(e) = cost(e) = f(z_v - z_u) + g(other\ factors)$ | In this equation, *zv*−*zu*represents the elevation change between nodes *u* and *v*, and *f* and *g* are functions that combine the elevation change with other relevant surface parameters. These functions can be application-specific and may involve factors like slope, terrain roughness, or energy consumption. The exact equations for *f*(*zv*−*zu*) and (other factors)*g*(other factors) would depend on the specific surface modeling, application, and desired cost functions. |
| 16.Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization[33]. | 2020 | Dijkstra and Bellman-Ford algorithms for shortest path optimization | relaxing the edge (*u*,*v*) is: $d(v) = \min\,(d(v), d(u) + w(u,v))$ | The Bellman-Ford Algorithm is used to discover the shortest path in a weighted network from a source node to all other nodes, even if the graph contains negative-weight edges or has a negative-weight cycle. It detects negative-weight cycles.<br>This equation updates the distance to node *v* by taking the minimum between its current distance ($d(v)$) and the sum of the distance to node *u* ($d(u)$) and the weight of the edge (*u*,*v*) $w(u,v)$). It ensures that the shortest path to *v* is considered. |
| 17. Optimal Route Planning of Parking Lot Based on Dijkstra Algorithm[34]. | 2017 | optimal route planning in parking lots using Dijkstra Algorithm | tentative distance *alt* is: $alt = d(u) + w(u,v)$ | This equation represents the sum of the current shortest distance to node *u* ($d(u)$) and the weight of the edge from *u* to its neighbouring node *v* ($w(u,v)$). If *alt* is smaller than the current shortest distance to *v* ($d(v)$), it means that there is a shorter path from the starting point to *v* through *u*, so $d(v)$ is updated with this shorter distance.<br><br>In the context of parking lots, the weight $w(u,v)$ may represent the time it takes to travel between parking spaces, considering factors such as distance, congestion, or even parking availability |
| 18. Dijkstra algorithm interactive training software development for network analysis applications in GIS[35]. | 2021 | GIS and Graph on Dijkstra's Algorithm using self-designed graphs. | Dijkstra's Algorithm in GIS (Geographic Information Systems) typically involves representing nodes (locations) and edges (connections) with associated weights (distances or costs).<br>**NODES: A, H , S , M**<br>**Edges (with distances):**<br>**- A to H: 5 km**<br>**- A to S: 8 km**<br>**- H to S: 2 km**<br>**- H to M: 6 km**<br>**- S to M: 4 km**<br>• The distance from Airport (A) to Hospital (H) is 5 kilometres.<br>• The distance from Airport (A) to School (S) is 8 kilometres. | This graph represents a simple geographical network where each node represents a location, and each edge represents a road or path between two locations with the associated distance.<br>For example, if you want to find the shortest path from the Airport (A) to the Mall (M), you would apply Dijkstra's Algorithm to this graph, and it would yield the path with the minimum total distance (in this case, A > H > S > M, with a total distance of 13 kilometres). |

| | | | | |
|---|---|---|---|---|
| | | | • The distance from Hospital (H) to School (S) is 2 kilometres[36].<br>• The distance from Hospital (H) to Mall (M) is 6 kilometres.<br>• The distance from School (S) to Mall (M) is 4 kilometres. | |
| 19.Algorithm To Find Tourism Place Shortest Route[37]: | 2012 | importance of algorithms in finding the shortest route for tourism places and factors | minimize total travel distance:<br>$$\sum_{i=1}^{n}\sum_{j=1}^{n} dij.xi.xj$$<br>Subject to:<br>• $\sum_{i=1}^{n} xi = n$ (Each attraction is visited exactly once)<br>• $\sum_{i=1}^{n} ti.xi \leq T$(Total travel time does not exceed the time constraint)<br>• $x \in \{0,1\}$ (Decision variable: 1 if attraction i is visited, 0 otherwise) | The actual optimization of the objective function while satisfying constraints is performed using optimization algorithms. Common algorithms include Genetic Algorithms, Ant Colony Optimization, and Mixed-Integer Linear Programming, among others. These algorithms iteratively adjust the decision variables to find the optimal route. This is a simplified example, and the actual formulation would depend on the specific constraints and objectives of the tourism route optimization problem. You may need to adapt and customize the equations and constraints to suit your particular scenario. Additionally, solving such optimization problems often requires specialized optimization software or libraries. |
| 20.An Extensive Review of Shortest Path Problem Solving Algorithms[38]. | 2021 | Shortest Path Algorithms (SPAs) | The equation for updating the shortest distance *d*[*v*] from the source node *s* to node *v* using Dijkstra's Algorithm is as follows<br>$$d(v) = \min\big(d(v), d(u) + w(u,v)\big)$$<br>• d[v]is updated if a shorter path to node *v* is found through node *u*.<br>• d[u]is the current shortest distance to node *u*.<br>• w(u,v) is the weight of the edge (*u*,*v*). | The algorithm iteratively selects the node with the minimum *d*[*v*] value among unvisited nodes and updates the distances until all nodes have been visited or the destination node is reached.<br>Let's define the following terms:<br>*V*: Set of nodes (vertices).<br>*E*: Set of edges, where each edge (*u*,*v*) has a non-negative weight *w* (*u*,*v*).<br>*s*: Source node.<br>*d*[*v*]: Shortest distance from the source node *s* to node *v*. |

## V. RESEARCH GAP

While Dijkstra's Algorithm is a powerful method for finding the shortest path in a weighted graph, there are still some research gaps and areas where further investigation is warranted, especially in the context of its application to the Transport Problem and related logistics and supply chain management challenges. Some potential research gaps include:
Scalability: Investigating methods to improve the scalability of Dijkstra's Algorithm for large-scale transportation networks, considering the computational complexity and memory requirements when dealing with a vast number of nodes and edges.
Real-time Optimization: Developing real-time optimization strategies that can dynamically adapt to changing transportation conditions, such as traffic congestion, road closures, or demand fluctuations, to provide more accurate and responsive routing solutions.
Multi-Objective Optimization: Extending Dijkstra's Algorithm to handle multi-objective optimization problems in logistics, where multiple criteria, such as cost, time, and environmental impact, need to be simultaneously considered when determining the optimal transportation routes.
Uncertainty Modelling: Integrating uncertainty modelling into Dijkstra's Algorithm for more robust decision-making in logistics and supply chain management. This includes handling uncertain travel times, demand variations, and other stochastic factors.
Advanced Data Sources: Leveraging emerging data sources, such as real-time GPS data, IoT sensors, and satellite imagery, to enhance the accuracy and reliability of input data for Dijkstra's Algorithm in transportation optimization.

Key Findings:

Despite these research gaps, several key findings highlight the practical significance of applying Dijkstra's Algorithm in addressing transportation and logistics challenges:

1. Efficiency: Dijkstra's Algorithm is highly efficient in finding the shortest path in weighted graphs, making it suitable for real-time or near-real-time routing and logistics optimization.
2. Flexibility: The algorithm's flexibility allows it to be applied to various transportation scenarios, including road networks, supply chain logistics, and even surface path planning.
3. Integration: Dijkstra's Algorithm can be integrated into larger optimization frameworks, such as linear programming models, to solve complex transportation optimization problems efficiently.
4. Adaptability: By considering different weight criteria (e.g., distance, time, cost), Dijkstra's Algorithm can adapt to diverse logistics and supply chain requirements.
5. Practical Use Cases: The algorithm has been effectively implemented in a variety of real-world circumstances, including route planning for delivery trucks, network optimization in telecommunications, and surface path planning for autonomous vehicles

## VI. CONCLUSION:

Dijkstra's Algorithm, originally developed for finding the shortest path in weighted graphs, has proven to be a versatile and powerful tool with applications in various fields, particularly in addressing transportation and logistics challenges. In this comprehensive overview, we have explored the core concepts of Dijkstra's Algorithm and its application in solving the Transport Problem.

Dijkstra's Algorithm, developed by Edsger W. Dijkstra in 1956, is a fundamental method for finding the shortest path in weighted networks. Its flexibility allows it to adapt to different criteria such as distance, time, or cost, making it valuable for solving complex problems in domains like network routing, logistics, and transportation optimization.Dijkstra's Algorithm operates through a systematic process that involves initializing distances, maintaining a priority queue, extracting nodes with the shortest known distances, relaxing edges, and iteratively updating distances until all nodes are visited or a target node is reached. This process efficiently determines optimal paths from a source node to all other nodes in the network.

Dijkstra's Algorithm can be adapted to efficiently solve the Transport Problem, where it finds the optimal routes for transporting goods from suppliers to consumers while minimizing transportation costs. The algorithm identifies the shortest paths and transportation costs, which can then be used in linear programming models to optimize logistics. We conducted a literature survey to explore various applications of Dijkstra's Algorithm and related mathematical models. These applications ranged from hazard material transportation and trade cost estimation to time-dependent shortest path algorithms and robust shortest path problems. Dijkstra's Algorithm plays a crucial role in solving these diverse problems, often integrated with other techniques and models to address real-world challenges.

While Dijkstra's Algorithm is highly efficient and flexible, there are research gaps that invite further exploration. These include enhancing scalability for large networks, enabling real-time adaptability, handling multi-objective optimization, modelling uncertainty, and integrating advanced data sources. Addressing these challenges will further enhance the algorithm's applicability in complex logistics and transportation scenarios. In conclusion, Dijkstra's Algorithm remains a cornerstone in solving transportation and logistics problems, offering an efficient and adaptable means of determining optimal routes. Its enduring significance is evident in applications spanning from supply chain management to route planning for autonomous vehicles, making it an invaluable asset in addressing real-world challenges in the ever-evolving field of transportation and logistics.

## REFERENCE

[1] M. A. Alam and M. O. Faruq, "Finding Shortest Path for Road Network Using Dijkstra's Algorithm," Bangladesh J. Multidiscip. Sci. Res., vol. 1, no. 2, pp. 41–45, 2019, doi: 10.46281/bjmsr.v1i2.366.

[2] B. Amaliah, C. Fatichah, and O. Riptianingdyah, "Finding the shortest paths among cities in Java Island using node combination based on Dijkstra algorithm," Int. J. Smart Sens. Intell. Syst., vol. 9, no. 4, pp. 2219–2236, 2016, doi: 10.21307/ijssis-2017-961.

[3] A. Shaikh and A. Dhale, "AGV Path Planning and Obstacle Avoidance Using Dijkstra's Algorithm," Int. J. Appl. or Innov. Eng. Manag., vol. 2, no. 6, pp. 77–83, 2013.

[4] R. Likaj, A. Shala, M. Mehmetaj, P. Hyseni, and X. Bajrami, "Application of graph theory to find optimal paths for the transportation problem," IFAC Proc. Vol., vol. 15, no. PART 1, pp. 235–240, 2013, doi: 10.3182/20130606-3-XK-4037.00031.

[5] O. Khaing, D. H. H. Wai, and D. E. E. Myat, "Using Dijkstra's Algorithm for Public Transportation System in Yangon Based on GIS," Int. J. Sci. Eng. Appl., vol. 7, no. 11, pp. 442–447, 2018, doi: 10.7753/ijsea0711.1008.

[6] N. Akpofure and N. Paul, "Anapplication of Dijkstra's Algorithm to shortest route problem," IOSR J. Math., vol. 13, no. 1, pp. 20–32, 2017, doi: 10.9790/5728-1303012032.

[7] R. L. Sah, "Dijkstra ' S Algorithm for Determining Shortest Path," vol. 7, no. 4, pp. 677–681, 2020.

[8] Z. Fuhao and L. Jiping, “An algorithm of shortest path based on Dijkstra for huge data,” 6th Int. Conf. Fuzzy Syst. Knowl. Discov. FSKD 2009, vol. 4, pp. 244–247, 2009, doi: 10.1109/FSKD.2009.848.
[9] R. P. S. Y. M. RK Arjun, “Research on the Optimization of Dijkstra’S Algorithm and Its Applications,” Int. J. Sci. Technol. Manag., vol. No 04, no. No. 01, April 2015, pp. 304–309, 2015, [Online]. Available: www.ijstm.com
[10]S. Kumawat, C. Dudeja, and P. Kumar, “An extensive review of shortest path problem solving algorithms,” Proc. - 5th Int. Conf. Intell. Comput. Control Syst. ICICCS 2021, no. Iciccs, pp. 176–184, 2021, doi: 10.1109/ICICCS51141.2021.9432275.
[11]E. D. Madyatmadja, H. Nindito, R. A. Bhaskoro, A. V. D. Sano, and C. P. M. Sianipar, “Algorithm to find tourism place shortest route: A systematic literature review,” J. Theor. Appl. Inf. Technol., vol. 99, no. 4, pp. 787–796, 2021.
[12]H. Yuan, J. Hu, Y. Song, Y. Li, and J. Du, “A new exact algorithm for the shortest path problem: An optimized shortest distance matrix,” Comput. Ind. Eng., vol. 158, no. August 2020, p. 107407, 2021, doi: 10.1016/j.cie.2021.107407.
[13]M. M. Ahmed, A. R. Khan, M. S. Uddin, and F. Ahmed, “A New Approach to Solve Transportation Problems,” Open J. Optim., vol. 05, no. 01, pp. 22–30, 2016, doi: 10.4236/ojop.2016.51003.
[14]M. L. Aliyu, U. Usman, Z. Babayaro, and M. K. Aminu, “A Minimization of the Cost of Transportation,” Am. J. Oper. Res., vol. 2019, no. 1, pp. 1–7, 2019, doi: 10.5923/j.ajor.20190901.01.
[15]S. X. Wang, “The improved Dijkstra’s shortest path algorithm and its application,” Procedia Eng., vol. 29, pp. 1186–1190, 2012, doi: 10.1016/j.proeng.2012.01.110.
[16]Y. D. Rosita, E. E. Rosyida, and M. A. Rudiyanto, “Implementation of dijkstra algorithm and multi-criteria decision-making for optimal route distribution,” Procedia Comput. Sci., vol. 161, pp. 378–385, 2019, doi: 10.1016/j.procs.2019.11.136.
[17]A. Trivella, F. Corman, D. F. Koza, and D. Pisinger, “The multi-commodity network flow problem with soft transit time constraints: Application to liner shipping,” Transp. Res. Part E Logist. Transp. Rev., vol. 150, no. April, p. 102342, 2021, doi: 10.1016/j.tre.2021.102342.
[18]V. B. S. de Zaldivar, “Reference (1).pdf,” Revista Europea de Estudios Latinoamericanos y del Caribe, vol. 95. pp. 71–95, 2013.
[19]M. Izdebski, I. Jacyna-Gołda, and P. Gołda, “Minimisation of the probability of serious road accidents in the transport of dangerous goods,” Reliab. Eng. Syst. Saf., vol. 217, no. June 2021, 2022, doi: 10.1016/j.ress.2021.108093.
[20]S. A. Bęczkowska and I. Grabarek, “The importance of the human factor in safety for the transport of dangerous goods,” Int. J. Environ. Res. Public Health, vol. 18, no. 14, 2021, doi: 10.3390/ijerph18147525.
[21]A. Idri, M. Oukarfi, A. Boulmakoul, K. Zeitouni, and A. Masri, “A new time-dependent shortest path algorithm for multimodal transportation network,” Procedia Comput. Sci., vol. 109, pp. 692–697, 2017, doi: 10.1016/j.procs.2017.05.379.
[22]C. Dong and R. Franklin, “From the Digital Internet to the Physical Internet: A Conceptual Framework With a Stylized Network Model,” J. Bus. Logist., vol. 42, no. 1, pp. 108–119, 2021, doi: 10.1111/jbl.12253.
[23]G. Yu and J. Yang, “On the robust shortest path problem,” Comput. Oper. Res., vol. 25, no. 6, pp. 457–468, 1998, doi: 10.1016/S0305-0548(97)00085-3.
[24]P. Kolman, P. Zach, and J. Holoubek, “The development of e-learning applications solving problems from graph theory,” Acta Univ. Agric. Silvic. Mendelianae Brun., vol. 61, no. 7, pp. 2311–2316, 2013, doi: 10.11118/actaun201361072311.
[25]F. Chen, Y. C. Wang, B. Wang, and C. C. J. Kuo, “Graph representation learning: A survey,” APSIPA Trans. Signal Inf. Process., vol. 9, 2020, doi: 10.1017/ATSIP.2020.13.
[26]F. de Soyres, A. Mulabdic, S. Murray, N. Rocha, and M. Ruta, “How much will the Belt and Road Initiative reduce trade costs?,” Int. Econ., vol. 159, no. July, pp. 151–164, 2019, doi: 10.1016/j.inteco.2019.07.003.
[27]R. Hassin, “Approximation Schemes for the Restricted Shortest Path Problem,” Math. Oper. Res., vol. 17, no. 1, pp. 36–42, 1992, doi: 10.1287/moor.17.1.36.
[28]M. Davoodi and M. Ghaffari, “Shortest path problem on uncertain networks: An efficient two phases approach,” Comput. Ind. Eng., vol. 157, no. February, p. 107302, 2021, doi: 10.1016/j.cie.2021.107302.
[29]C. Italia, “Stefano PALLOTTI NO,” vol. 26, pp. 38–64, 1986.
[30]J. Tarbouriech, R. Zhou, S. S. Du, M. Pirotta, M. Valko, and A. Lazaric, “Stochastic Shortest Path: Minimax, Parameter-Free and Towards Horizon-Free Regret,” Adv. Neural Inf. Process. Syst., vol. 9, no. NeurIPS, pp. 6843–6855, 2021.
[31]D. Ding and X. Zou, “The Optimization of Logistics Distribution Route Based on Dijkstra’s Algorithm and C-W Savings Algorithm,” no. Mmebc, 2016, doi: 10.2991/mmebc-16.2016.200.
[32]M. Luo, X. Hou, and J. Yang, “Surface Optimal Path Planning Using an Extended Dijkstra Algorithm,” IEEE Access, vol. 8, pp. 147827–147838, 2020, doi: 10.1109/ACCESS.2020.3015976.
[33]S. W. G. Abusalim, R. Ibrahim, M. Zainuri Saringat, S. Jamel, and J. Abdul Wahab, “Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization,” IOP Conf. Ser. Mater. Sci. Eng., vol. 917, no. 1, 2020, doi: 10.1088/1757-899X/917/1/012077.
[34]Yujin and G. Xiaoxue, “Optimal Route Planning of Parking Lot Based on Dijkstra Algorithm,” Proc. - 2017 Int. Conf. Robot. Intell. Syst. ICRIS 2017, pp. 221–224, 2017, doi: 10.1109/ICRIS.2017.62.
[35]I. R. Karaş, O. Yasin, and M. K. Turan, “Interactive training software for optimum travel route analysis applications in railway networks,” vol. 16, no. 2, pp. 81–87, 2013.
[36]“Algorithms problem statement.pdf.”
[37]A. Fitriansyah, N. W. Parwati, D. R. Wardhani, and N. Kustian, “Dijkstra’s Algorithm to Find Shortest Path of Tourist Destination in Bali,” J. Phys. Conf. Ser., vol. 1338, no. 1, pp. 1163–1168, 2019, doi: 10.1088/1742-6596/1338/1/012044.
[38]A. Madkour, W. G. Aref, F. U. Rehman, M. A. Rahman, and S. Basalamah, “A Survey of Shortest-Path Algorithms,” pp. 1–26, 2017, [Online]. Available: http://arxiv.org/abs/1705.02044