International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 12, December 2022

Optimization of Dijkstra's Algorithm by Reducing the Number of Iterations

Chirag Desai
Faculty
Information Technology
K. J. Somaiya College of Engineering

Param Mamania
Student
Information Technology
K. J. Somaiya College of Engineering

Arya Karambelkar
Student
Information Technology
K. J. Somaiya College of Engineering

***Abstract*—In this paper, we propose a change to Dijkstra's algorithm that minimizes the number of iterations while improving the method's efficiency. In the traditional Dijkstra's technique, the core principle is to address the problem when more than one node passes the second step requirement. After incorporating the proposed changes, the maximum number of iterations of Dijkstra's method is fewer than the number of nodes in the graph.**

Keywords—Dijkstra's algorithm, directed graph, shortest path

I. INTRODUCTION

The shortest-route problem determines the shortest path in a weight graph (digraph) in a transportation network between two given vertices, source and destination. Other scenarios, such as VLSI design and equipment replacement, can be represented by the same paradigm. To discover the shortest path in any graph, there are several types of shortest path algorithms. Most frequently encountered are the following:

- Shortest path between two specified vertices
- Shortest path between all pairs of vertices
- Shortest path from a specified vertex to all others

The Dijkstra algorithm finds the shortest paths between nodes in a network. Edsger W. Dijkstra, a computer scientist, devised it in 1956 and published it in 1959.

The Dijkstra algorithm finds the shortest paths between nodes in a network. Edsger W. Dijkstra, a computer scientist, devised it in 1956 and published it in 1959.

It can also be used to find the shortest paths from a single source node to a single destination node, with the process stopping once the shortest path is found. Dijkstra's algorithm, for example, can be used to find the shortest route between one city and all other cities if the nodes of the graph represent cities and the edge path costs represent driving distances between pairs of cities connected by a direct road (ignoring red lights, stop signs, tolls, and other obstructions for simplicity). Shortest route algorithms are commonly used in network routing protocols such as IS-IS (Intermediate System to Intermediate System) and OSPF (Open Shortest Path First). It is also used as a subroutine in algorithms like Johnson's.

As labels, the Dijkstra algorithm uses entirely sorted positive integers or real values. It may be expanded to use any partially ordered labels as long as the subsequent labels (generated while traversing an edge) are monotonically non-decreasing. The Dijkstra generalized shortest-path algorithm is the name given to this generalization.

The Dijkstra algorithm is improved in terms of efficient implementation and cost matrix. In this paper, we propose some improvements to reduce the number of iterations and to find the shortest path easily and quickly.

II. LITERARY REVIEW

1. Hwan Il Kang et al. created a path planning method that makes use of an upgraded Dijkstra algorithm with particle swarm optimization. To find the best path, they first built the maklink on the global environment and then created a graph connected with the maklink. They derive the Dijkstra route from the graph.
2. Sidharth Parekh et. al developed an exhaustive Approach Orchestrating Negative Edges for Dijkstra's Algorithm.
3. Omoniyi Ajoke Gbadamosi et. al Designed a Modified Dijkstra's Algorithm for finding alternate routes for shortest-path problems with huge costs.
4. Kaicong Wei et. al introduced the Dijkstra's algorithm for solving the problem of finding the shortest path, and the Dijkstra's algorithm is modified to solve the problem of finding the maximum load path.

III. METHODOLOGY

A. Dijkstra's Algorithm

The challenge of determining the shortest route from a given vertex s to mother t may be expressed as follows:
A n by n matrix D=[dij] describes a basic weighted digraph G with n vertices: dij = length (or distance or weight) of the directed edge from vertex I to vertex j:

$$d_{ij} = \begin{cases} > 0, \ if \ i \neq j \\ = 0, if \ i = j \\ = \infty \ (large \ number), if \ there \ is \ no \ edge \ from \ i \ to \ j \end{cases}$$

Dijkstra's algorithm labels the vertices of a given digraph, with some vertices having permanent labels and others having temporary labels at each stage of the algorithm. The algorithm begins by assigning the starting vertex s a permanent label of 0 and the remaining n-1 vertices a temporary label of infinity.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 12, December 2022

In subsequent iterations, another vertex assigns a permanent label based on the following rules:

a) Every unlabeled vertex j is assigned a new temporary label with the value min[old label of j, (old label of I + dij)], where I is the most recently permanently labeled vertex in the previous iteration and dij is the direct distance between I and j. Dij = infinite if I and j are not joined by an edge..
b) The temporary label with the lowest detected value is selected as the permanent label for the relevant vertex. If there is more than one shortest route, choose any of the candidates for permanent labeling. Steps a and b are continued alternately until the target vertex t is permanently labeled. The first vertex that has been permanently marked is zero distance from s. The vertex closest to s is the second to be permanently labeled (among the remaining n-1 vertices). From the remaining n2 vertices, the second closest vertex to s is the next to be permanently labeled. So on and so on. Each vertex's permanent label is the vertex's shortest distance from s. This proposition might be established via induction.

B. Modified Dijkstra's Algorithm

In this section, we will present a modified version of Dijkstra's Algorithm to reduce the total number of iterations by optimizing the situation of many shortest paths:

a) Every unlabeled vertex j gets a new temporary label with the value min[old label of j, (old label of I + dij)], where I is the most recently permanently labeled vertex in the previous iteration and dij is the direct distance between vertices I and j. Dij=infinity if I and j are not linked by an edge.
b) The temporary label with the lowest detected value is selected as the permanent label for the relevant vertex. Choose all of the shortest pathways for permanent labeling if there are numerous.
c) Steps (a) and (b) are repeated alternately maximum n-1 times until the destination vertex t gets a permanent label.
d) Le T=[tij] is the shortest path matrix that we can build it form D, as follows

$$t_{ij} = \begin{cases} 0, if\ d_{ji} = 0\ or\ d_{ji} = \infty \\ d_{ji}, otherwise \end{cases}$$

IV. COMPARISON

The network in the following figure (figure 1) gives the distances in miles between pairs of cities 1,2,… and 8. For efficiency purpose, we will find the shortest route between cities 1 and 8 using the traditional and modified version of Dijkstra algorithm:

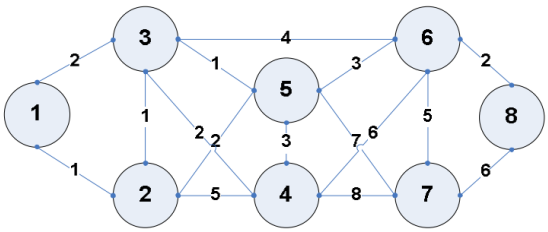


Fig. 1: Network of 8 nodes

Dijkstra algorithm: The number of iterations to find the shortest route between cities 1 and 8 by using Dijkstra algorithm is 8

Modified Dijkstra algorithm: By using the modified version of Dijkstra algorithm, 5 iterations only are required to get the shortest route between cities 1 and 8

Initially

$$D = \begin{bmatrix} 0 & 1 & 2 & \infty & \infty & \infty & \infty & \infty \\ 1 & 0 & 1 & 5 & 2 & \infty & \infty & \infty \\ 2 & 1 & 0 & 2 & 1 & 4 & \infty & \infty \\ \infty & 5 & 2 & 0 & 3 & 6 & 8 & \infty \\ \infty & 2 & 1 & 3 & 0 & 3 & 7 & \infty \\ \infty & \infty & 4 & 6 & 3 & 0 & 5 & 2 \\ \infty & \infty & \infty & 8 & 7 & 5 & 0 & 6 \\ \infty & \infty & \infty & \infty & \infty & 2 & 6 & 0 \end{bmatrix}$$

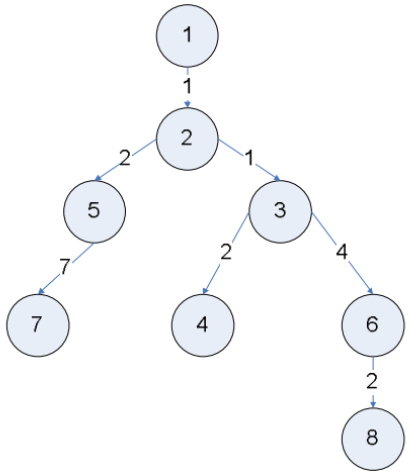
:

and the set of permanent vertices is P={1}.

Iteration 1:	P = { 1,2 }, d1 j = { 0,1,2,4 ∞,∞,∞,∞}
Iteration 2:	P = { 1,2,3 }, d1 j = { 0,1,2,4,3,6, ∞,∞}
Iteration 3:	P = { 1,2,3,5 }, d1 j = { 0,1,2,4,3,6,10 ,∞}
Iteration 4:	P = { 1,2,3,4,5,6 }, d1 j = { 0,1,2,4,3,6,10,8}
Iteration 5:	P = { 1,2,3,4,5,6,7,8 }, d1 j = { 0,1,2,4,3,6,10,8 }

$$T = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 12, December 2022



Based on the above tree, the shortest route and distance between cities 1 and 8 are: 1-2-3-6-8 and the distance is 8 miles.

V. CONCLUSION

The Dijkstra algorithm has been optimized in this article. The major goal is to address the second stage of the traditional version of this approach, in which there are several shortest pathways between a node and its successors, and to reduce the number of repeats. The proposed update decreases the number of iterations greatly and simplifies the process of identifying the shortest route between any two cities. Future work will entail putting the recommended changes into action and comparing their complexity to the current system.

REFERENCES

[1] H. I. Kang, B. Lee and K. Kim, "Path Planning Algorithm Using the Particle Swarm Optimization and the Improved Dijkstra Algorithm," 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, 2008, pp. 1002-1004, doi: 10.1109/PACIIA.2008.376.

[2] S. Parekh, A. Jha, A. Dalvi and I. Siddavatam, "An Exhaustive Approach Orchestrating Negative Edges for Dijkstra's Algorithm," 2022 IEEE 7th International conference for Convergence in Technology (I2CT), 2022, pp. 1-5, doi: 10.1109/I2CT54291.2022.9824795.

[3] O. A. Gbadamosi and D. R. Aremu, "Design of a Modified Dijkstra's Algorithm for finding alternate routes for shortest-path problems with huge costs," 2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS), 2020, pp. 1-6, doi: 10.1109/ICMCECS47690.2020.240873.

[4] , and the Dijkstra's algorithm is modified to solve the problem of finding the maximum load path

[5] Mingjun Wei and Yu Meng, "Research on the optimal route choice based on improved Dijkstra," 2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA), 2014, pp. 303-306, doi: 10.1109/WARTIA.2014.6976257.

[6] L. Wenzheng, L. Junjun and Y. Shunli, "An Improved Dijkstra's Algorithm for Shortest Path Planning on 2D Grid Maps," 2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC), 2019, pp. 438-441, doi: 10.1109/ICEIEC.2019.8784487.