International Journal for Research in Engineering Application & Management (IJREAM)
ISSN : 2454-9150 Vol-05, Issue-07, Oct 2019

Shortest Path in Multi-stage Graph using Dynamic Programming approach

[1]S.P. Behera , [2]S.Bhattacharjee, [3]D.Mishra

[1]Ph.D Research Scholar, [2]Assistant Professor, [3]Professor

Department of Mathematics, Ravenshaw University, Cuttack, Odisha, India.

C.V Raman (Autonomous) Engineering College ,Bhubaneswar, Odisha, India.

Abstract: The shortest path problem is a classic problem in combinatorial optimization .In this paper , we propose Dynamic programming approach algorithm to find the shortest path from source to destination associated with multistage graph & effectiveness of the algorithm is explained with respect to various strategies like Brute force, Dijkstra's algorithm & Greedy method. We have also implemented the proposed algorithm to find the shortest path of multistage network using Java Programming language finally we have presented some numerical examples to explain the solution procedure.

Keywords: Multistage graph, Brute force strategy, Greedy approach, Dijkstra's algorithm, Dynamic programming

I. INTRODUCTION

Generally shortest path problem is a problem of finding the shortest path or route from a starting point to a final destination. Generally, in order to represent the shortest path problem we use graphs. A graph, which contains sets of vertices and edges. Edges connect pairs of vertices. Along the edges of a graph it is possible to walk by moving from one vertex to other vertices. Depending on whether or not one can walk along the edges by both sides or by only one side determines if the graph is a directed graph or an undirected graph. In addition, lengths of edges are often called weights, and the weights are normally used for calculating the shortest path from one point to another point. In the real world it is possible to apply the graph theory to different types of scenarios. For example, in order to represent a map we can use a graph, where vertices represent cities and edges represent routes that connect the cities. If routes are one-way then the graph will be directed; otherwise, it will be undirected. There exist different types of algorithms that solve the shortest path problem. However, only several of the most popular conventional shortest path algorithms along with one that uses genetic algorithm are going to be discussed in this paper, and they are as follows: 1.Dijkstra's Algorithm 2.Floyd-Warshall Algorithm 3. Bellman-Ford Algorithm 4. Genetic Algorithm (GA) Other than GA, nowadays, there are also many intelligent shortest path algorithms that have been introduced in several past research papers. For example, the authors in [1] used a heuristic method for computing the shortest path from one point to another point within traffic networks.

They proposed a new dynamic direction restricted algorithm obtained by extending the Dijkstra's algorithm [1].‖ in another paper [2], a heuristic GA was used for solving the single source shortest path (SSSP) problem. Its main goal was to investigate the SSSP problem within the Internet routing setting, particularly when considering the cost of transmitting messages/packets is significantly high, and the search space is normally very large [8].

In this paper we are going to find out shortest path in multistage graph using Dynamic programming approach. Which in turn will find out the shortest path a workflow can take to complete a task which is minimum distance taken in sense.

II. LITERATURE REVIEW

Dijkstra (1959) proposed a graph search algorithm that can be used to solve the single-source shortest path problem for any graph that has a non-negative edge path cost. This graph search algorithm was later modified by Lee in 2006 and was applied to the vehicle guidance system. This vehicle guidance system is divided into two paths; namely, the shortest path and the fastest path algorithms (Chen et al., 2009).While the shortest path algorithm focuses on route length parameter and calculates the shortest route between each order pair, the fastest path algorithm focuses on the path with minimum travel time. The future travel time can be predicted based on prediction models using historical data for link travel time information which can be daily, weekly or even a session. Meghanathan (2012) reviewed Dijkstra's algorithm and Bellman-Ford algorithm for finding the shortest path in a graph. He concluded that the time complexity of Dijkstra's algorithm is O (|E|*log |V|)

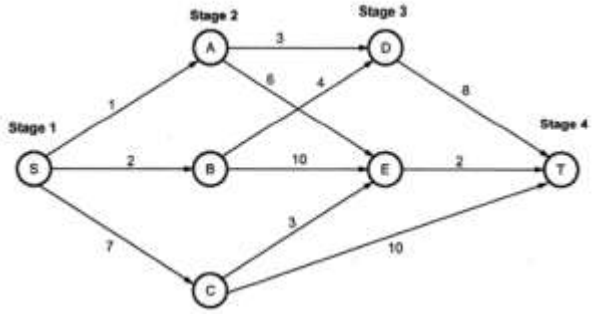
 DOI : 10.35291/2454-9150.2019.0470

while the time complexity of the Bellman-Ford algorithm is O(|V||E|). Approximate shortest paths between graph nodes using a collection of spanning trees. Spanning trees are easy to generate, compact relative to original graphs, and can be distributed across machines to parallelize queries. They demonstrate its scalability and effectiveness using large social graphs from Face book, Orkut, and Renren, the largest of which includes 43 million nodes and 1 billion edges. They describe techniques to incrementally update Atlas as social graphs change over time. They capture graph dynamics using 35 daily snapshots of a Face book network and show that Atlas can amortize the cost of tree updates over time. Finally, they apply Atlas to several graph applications and show that they produce results that closely approximate ideal results [8].

III. METHODOLOGY

The cost of a path from source (denoted by S) to sink (denoted by T) is the sum of the costs of edges on the path. In multistage graph problem we have to find the path from S to T. there is set of vertices in each stage. The multistage graph can be solved using forward and backward approach. Let us solve multistage problem for both the approaches with the help of example

Example1. Consider the graph G as shown in the figure



IV. EXPERIMENT

4.1 Result analysis with result to Brute force, Dijkstra's algorithm & Simple Greedy method:

- **Brute force** method of finding all possible paths between Source and Destination and then finding the minimum. That's the WORST possible strategy.
- **Dijkstra's Algorithm** of Single Source shortest paths. This method will find shortest paths from source to all other nodes which are not required in this case. So it will take a lot of time and it doesn't even use the SPECIAL feature that this MULTI-STAGE graph has.
- **Simple Greedy Method** – At each node, choose the shortest outgoing path. If we apply this approach to the example graph give above we get the solution as 1 + 3 + 8 = 12. But a quick look at the graph will show much shorter paths available than 12 So the greedy method fails

4.2 Result analysis with use of Dynamic Programming algorithm approach

The best option is Dynamic Programming. So we need to find Optimal Sub-structure, Recursive Equations and Overlapping Sub-problems. Optimal Substructure and Recursive Equation

There is a single vertex in stage 1, then 3 vertices in stage 2, then 2 vertices in stage 3 and only one vertex in stage 4 (this is a target stage).

4.2.1 Explanation of Dynamic programming approach:

We define the notation: - M(x, y) as the minimum cost to T (target node) from Stage x, Node y.

Shortest distance from stage 1, node 0 to

destination, i.e., 7 is M (1, 0).

// From 0, we can go to 1 or 2 or 3 to

// reach 7.

M (1, 0) = min (1 + M (2, 1),

2 + M (2, 2),

5 + M (2, 3))

This means that our problem of 0 —> 7 is now sub-divided into 3 sub-problems

So if we have total 'n' stages and target

*as T, then the **stopping condition** will be :-*

M(n-1, i) = i ---> T + M(n, T) = i ---> T

Recursion Tree and Overlapping Sub-Problems:-
So, the hierarchy of M(x, y) evaluations will look something like this :-

In M(i, j), i is stage number and

j is node number

M(1, 0)

*/ | *

*/ | *

M(2, 1) M(2, 2) M(2, 3)

*/ \ / \ / *

M(3, 4) M(3, 5) M(3, 4) M(3, 5) M(3, 6) M(3, 6)

. . . . . .

 DOI : 10.35291/2454-9150.2019.0470

So, here we have drawn a very small part of the Recursion Tree and we can already see Overlapping Sub-Problems. We can largely reduce the number of M(x, y) evaluations using Dynamic Programming

4.2.1.1 Backward approach :

d(S, T)=min {1+d(A, T),2+d(B,T),7+d(C,T)} .....(1)

Calculation of d(A,T), d(B,T) and d(C,T).

d(A,T)=min{3+d(D,T),6+d(E,T)}..... . (2)

d(B,T)=min{4+d(D,T),10+d(E,T)}..... (3)

d(C,T)=min{3+d(E,T),d(C,T)}..... (4)

Now let us compute d(D,T) and d(E,T).

d(D,T)=8

d(E,T)=2 backward vertex=E

Let us put these values in equations (2), (3) and (4)

d(A,T)=min{3+8, 6+2}

d(A,T)=8 A-E-T

d(B,T)=min{4+8,10+2}

d{B,T}=12 A-D-T

d(C,T)=min(3+2,10)

d(C,T)=5 C-E-T

d(S,T)=min{1+d(A,T), 2+d(B,T), 7+d(C,T)}

=min{1+8, 2+12,7+5}

=min{9,14,12}

d(S,T)=9 S-A-E-T

The path with minimum cost is S-A-E-T with the cost 9.

4.2.1.2 Forward approach

d(S,A)=1

d(S,B)=2

d(S,C)=7

d(S,D)=min{1+d(A,D),2+d(B,D)}

=min{1+3,2+4}

d(S,D)=4

d(S,E)=min{1+d(A,E), 2+d(B,E),7+d(C,E)}

=min {1+6,2+10,7+3}

=min {7,12,10}

d(S,E)=7 i.e. Path S-A-E is chosen.

d(S,T)=min{d(S,D)+d(D,T),d(S,E)+d(E,T),d(S,C)+d(C,T)}

=min {4+8,7+2,7+10}

d(S,T)=9 i.e. Path S-E, E-T is chosen.

The minimum cost=9 with the path S-A-E-T.

4.2.1.3 .Implementation using Java Programming:

```
class GFG
{

static int N = 8;
static int INF = Integer.MAX_VALUE;

// Returns shortest distance from 0 to
// N-1.
public static int shortestDist(int[][] graph)
{
// dist[i] is going to store shortest
// distance from node i to node N-1.
int[] dist = new int[N];

dist[N - 1] = 0;

// Calculating shortest path for
// rest of the nodes
for (int i = N - 2; i >= 0; i--)
{

// Initialize distance from i to
// destination (N-1)
dist[i] = INF;

// Check all nodes of next stages
// to find shortest distance from
// i to N-1.
for (int j = i; j < N; j++)
{

// Reject if no edge exists
if (graph[i][j] == INF)

continue;

// We apply recursive equation to
// distance to target through j.
// and compare with minimum distance
// so far.
dist[i] = Math.min(dist[i], graph[i][j]
+ dist[j]);
}
}
return dist[0];
}

// Driver code
public static void main(String[] args)
{
// Graph stored in the form of an
// adjacency Matrix
int[][] graph = new int[][]{{INF, 1, 2, 5, INF, INF, INF,
INF},
{INF, INF, INF, INF, 4, 11, INF, INF},
{INF, INF, INF, INF, 9, 5, 16, INF},
{INF, INF, INF, INF, INF, INF, 2, INF},
{INF, INF, INF, INF, INF, INF, INF, 18},
{INF, INF, INF, INF, INF, INF, INF, 13},
{INF, INF, INF, INF, INF, INF, INF, 2}};
System.out.println(shortestDist(graph));
}
}
```

 DOI : 10.35291/2454-9150.2019.0470

Output: *9*

So the path with minimum cost is S-A-E-T with the cost 9.

4.3 Time Complexity Analysis :

The time complexity for each algorithm is illustrated in Table I; n represents the total number of vertices, and m is the total number of edges

Algorithm	Time Complexity
Dijkstra	$n^2 + m$
Bellman-Ford	n^3
Floyd-Warshall	nm

But the time complexity by using Dynamic approach is $O(n^2)$, which is least time complexity among Dijkstra algorithm, Bellman-Ford & Floyd-Warshall algorithm.

V. CONCLUSIONS

The time complexity for each of the Dijkstra's, Floyd-Warshall and Bellman-Ford algorithms show that these algorithms are acceptable in terms of their overall performance in solving the shortest path problem. All of these algorithms produce only one solution. However, the main advantage of Dynamic approach for shortest path associated with multi–stage over these algorithms is that it may produce a number of different optimal solutions since the result can differ every time the it is executed. In the future our proposed frame work will be extended and improved in finding the shortest path or distance between two places in a map that represents any types of networks. In addition, other artificial intelligence techniques such as fuzzy logic and neural networks can also be implemented in improving existing shortest path algorithms in order to make them more intelligent and more efficient

REFERENCES

[1] C. Xi, F. Qi, and L. Wei, ―A New Shortest Path Algorithm based on Heuristic Strategy,‖ Proc. of the 6th World Congress on Intelligent Control and Automation, Vol. 1, pp. 2531–2536, 2006.

[2] B.S. Hasan, M.A. Khamees, and A.S.H. Mahmoud, ―A Heuristic Genetic Algorithm for the Single Source Shortest Path Problem, ‖ Proc. of International Conference on Computer Systems and Applications

[3] T. Li, L. Qi, and D. Ruan, ―An Efficient Algorithm for the Single-Source Shortest Path Problem in Graph Theory‖, Proc. of 3rd International Conference on Intelligent System and Knowledge Engineering, Vol. 1, pp. 152-157, 2008.

[4] M. Jordan, ―Notes 7 for CS170‖, UC Berkeley, 2005.

[5]. Ahmat, K. A. (n.d.). *Graph Theory and Optimization Problems for Very Large Networks.* New York: City University of New

York/Information Technology.

[6]. Brendan, H. a. (2005). Generalizing Dijkstra's Algorithm and Gaussian Elimination for solving MDPs. International Conference onAutomated Planning and Scheduling/Artificial Intelligence Planning System, (pp. 151 - 160).

[7]. Chen, K. M. (2009). A real-time wireless route guidance system for urban traffic management and its performance evaluation.

Papers of the 70th Vehicular Technology Conference Anchorage VTC, (pp. 1 -5).

[8]. Ebrahimnejad, S. A. (2011). Find the Shortest Path in Dynamic Network using Labelling Algorithm. Journal of Business and Social

Science.

[9]. Goldberg, A. V. (n.d.). Point - to - Point Shortest Path Algorithms with Pre-processing. Silicon Valley:MicrosoftResearch.

[10]. Goyal, S. (n.d.). A Survey on Travelling Salesman Problem. North Dakota: Department of Computer, University of North Dakota.

[11]. Lee, D. C. (2006). Proof of a modified Dijkstra's algorithm for computing shortest bundle delay in networks with deterministically

[12]. T. Li, Q. a. (2008). An efficient Algorithm for the single source Shortest Path Problem in Graph Theory. International Conference

on Intelligent System and Knowledge Engineering, (pp. 152 - 157).

[13]. Taha, H. (2011). Operations research an introduction, ninth edition. Pearson Publisher. Wadhwa, S. (n.d.). Analysis of a network design problem. 2000: Lehigh University

 DOI : 10.35291/2454-9150.2019.0470