Shortest Path Finding Algorithms for Real Road Network

Agam Mathur[1], Mayuresh Jakhotia[2], Anish Lavalekar[3], Nikita Magar[4]

[1,2,3,4]*Computer Department, VIIT, Pune University*

Abstract **-Several graph analysis algorithms for shortest path can be categorized as single source, single destination, or all pairs algorithms [4],[7]. Dijkstra's algorithm is one such which falls in the first category, and can be used to find the lowest cost between two nodes when there is a single source. However, it doesn't allow negative weights. It may simply fail in those cases to give correct results. But, in our system, we need to find the shortest path between on real road networks. There are several algorithms for this purpose, such as Bellman-Ford, A*, and Floyd-Warshall. In Bellman-Ford, the working is similar to Dijkstra's, i.e. it's a single source shortest path problem [4], but, it can handle negative weight edges. In the Floyd-Warshall algorithm, every node can be a source, and can be used to calculate the shortest distance from a source to a destination node. Unlike Dijkstra's algorithm, Floyd-Warshall, in case of negative cycles, will correctly state that there isn't any minimum weight path, owing to the unbounded negative weight. In short, Floyd-Warshall is an appropriate algorithm for our system which will be used in generating optimal route for all node pairings.**

Keyword -Travelling Salesperson Problem, Time complexity, Space complexity, Successor node, Intermediate path, Heuristic function.

I.INTRODUCTION

Graph theory has become an important means to represent and analyze various practical problems. In this context, a graph is made up of vertices/nodes, and lines called edges that connect the nodes. We are using short- est path algorithms in this paper, in order to come up with a solution for our system. Basically, the categories of any shortest path algorithm can broadly fall into single source, single destination, or all-pairs shortest path algorithms. The single source algorithm can be used to find the shortest path from a source vertex to all the other vertices in a graph. The single destination algorithm is used to find the shortest path in a directed graph, from all vertices to a single destination. It can be also be seen as a single source problem by reversing the arcs. The last category, all-pairs vertices, finds shortest path between every pair of vertices. The single source problems involve a single source and can be solved using algorithms such as Dijkstra's, and Bellman-Ford if negative edges are involved. A* algorithm is a single pair and uses heuristics to speed up the search. All-pair algorithms such as Johnson's algorithm and Floyd-Warshall algorithm are used to find shortest paths between all pairs of vertices. Johnson's al- gorithm uses Bellman-Ford and Dijkstra's algorithm, and is not efficient in dense graphs. Floyd-Warshall, on the other hand, is more useful in the case of dense graphs, such as real road networks. Our system is a public conveyance system wherein real road networks will be used. So our graph would be a dense graph, consisting of all the places as nodes, and the paths connecting the nodes, as the edges. Here, we are required to find out node to node path, and not a path from a single source to all the other nodes. Therefore, single source algorithms such as Dijkstra's and Bellman-Ford cannot be used in this case. Also, since our graph is not a sparse graph, we would not use Johnson's algorithm. Floyd-Warshall, on the other hand, provides the node to node distance as is required for our system.

II.ALGORITHM ANALYSIS

A.Dijkstra's algorithm:

Dijkstra's algorithm [1],[7]is a single source-single destination algorithm used to find the shortest path. By single source algorithm, we mean that there is shortest path between one source and multiple destinations. Single destination means that there is shortest path between multiple sources connecting a single destination. This can be seen as a single source shortest path algorithm by reversing the edges. So, basically, Dijkstra's algorithm is a single source shortest path problem, producing a shortest path tree for a graph with positive edge path costs.

In Dijkstra's algorithm, for a given vertex, i.e. node in the graph, the algorithm finds path with the lowest cost between that and all other vertices. It's also used to find cost of shortest path from a single source to a single destination by stopping the algorithm once the shortest path to the destination vertex has been determined.
Dijkstra's original algorithm runs in time O |V| , where |V| is the total number of vertices. Dijkstra's algorithm is asymptotically the fastest known single-source shortest path algorithm for arbitrary directed graphs having unbounded non-negative weights. Dijkstras algorithm is more general, it is not just restricted to acyclic graphs. The edges need not be investigated often using this algorithm. It means that once it has been carried out, the least path to all permanently labeled nodes can be found out without the need of a new diagram for each pass. It implies that it would turn out faster, if edges are relatively expensive to compute. Also, the order of $|V|_2$ indicates that it is efficient enough for relatively large problems. However, it requires that the weights on the edges must be positive and doesn't support negative edges. Also, Dijkstras algorithm considers only the weights between the vertices to select the shortest path. But, for situations like real road networks, there can be various parameters such as the time taken to travel (in case of a bus), the congestion on the roads, the monetary cost, etc. This

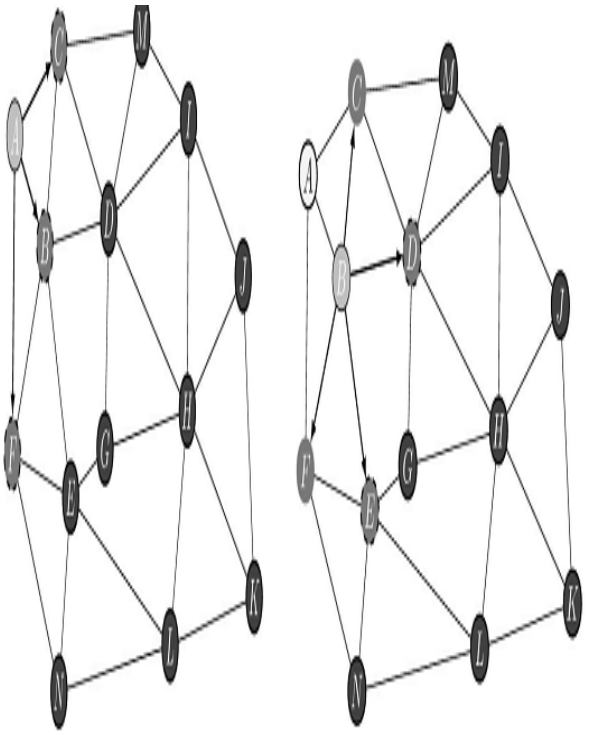

Figure 1: The first two steps of Dijkstras shortest path algorithm, displaying its breadth-first search properties

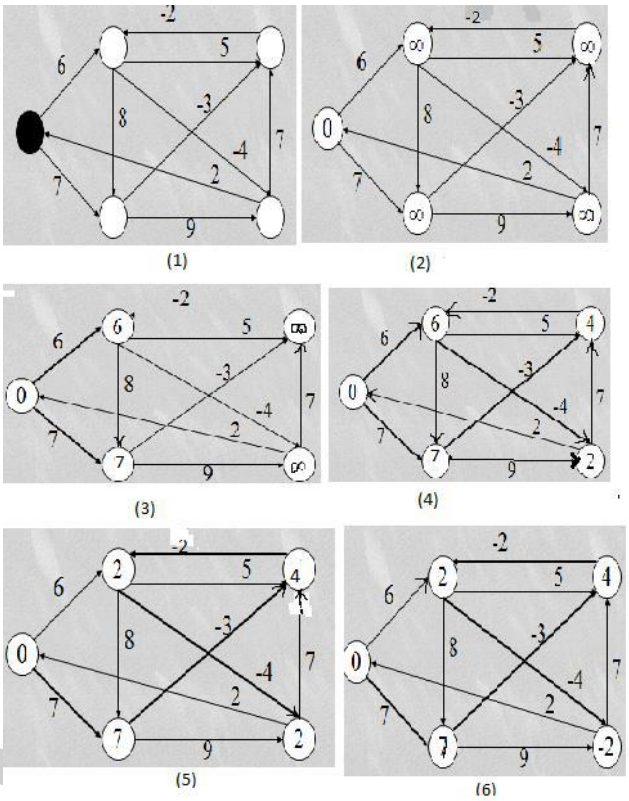

Figure 2: Working of Bellman-ford algorithm

Algorithm doesn't suffice any such parameters and so, is limited in its application from such system(s) point of view.

B. Bellman-Ford Algorithm

Bellman-Ford algorithm[5] is a single source algorithm used to find the shortest path in a weighted directed graph. By single source algorithm, we mean that there is shortest path between one source and multiple destina- tions. However, unlike Dijkstras, it considers even nega- tive edge weights. In this algorithm, if a graph contains a cycle with edges summing up to a negative value, then low value paths can be created. Bellman-Ford runs in O(|V|.|E|), where |V| is the total number of vertices, and | E| is the total number of edges.This algorithm simply relaxes all the edges until |V|-1 is reached. With the repe- titions, the number of vertices having correctly calculated distances grows, thereby all vertices will have correct distances. This is one of the reasons for the application of this algorithm to a wider class of inputs. It also in- creases system performance by allowing splitting of traffic across several paths, and works well for distributed systems. This algorithm does not scale well. Network topology can be changed but the changes made arenot reflected quickly due to the spreading of updates from node to node. Also, thereo s the count to infinity problem wherein if a node failure occurs, and the node is rendered unreachable from some other nodes, then those nodes may spend forever increasing their estimates to reach the node which has failed.

C. A Algorithm*

A* algorithm is an extension of Dijkstras algorithm de- scribed above. A* algorithm is used in pathfinding and efficient graph traversal between nodes. A* algorithm has better time complexity than Dijkstras algorithm by us- ing a heuristic function because of which it can guide the search to the desired solution and hence less time complexity.

A* algorithm uses Breadth-First-Search strategy to find the least cost path from a starting node to a final node. A* uses the cost function f(n) to sort the alternate path segments in a priority queue. Cost function f(n) is a sum of two functions g(n) and h(n). g(n) is the actual cost from starting node to current node n, while h(n) is the heuristic function which is the estimated cost of the cheapest path from current node n to final goal node. f(n) = g(n) + h(n)

A* algorithm accuracy and reduction in time complex- ity depends largely on the heuristic function used. For a given problem, many heuristic functions are possible and they will give different results. For example 8-puzzle game problem can be solved using two heuristics. Firstly

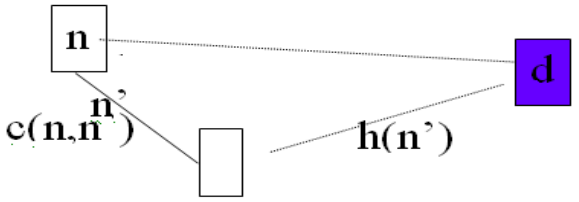

Figure 3: Consistent property of Heuristic Function

by "number of misplaced tiles in board n" and secondly by "sum of distances of misplaced tiles to goal positions in board n". Research shows that A* algorithm which uses second heuristic functions performs 10 times better than A* algorithm implemented by first heuristic.

Heuristic function h(n) used in A* algorithm should be admissible and consistent for generating optimal results. Heuristic function h(n) is admissible if it never overestimates the cost to reach the final destination node from the current node. In other words, heuristic function h(n) should be too optimistic, that it estimates the cost to be smaller than it actually is.

Heuristic function h(n) is consistent if for every node n and for every successor node nof n satisfies:

$h(n) <= c(n,n) + h(n)$

D. Floyd-Warshall Algorithm

The Floyd−Warshall algorithm [2],[4],[5],[7] (also known as Floyd's algorithm, Roy−Warshall algorithm, Roy−Floyd algorithm, or the WFI algorithm).This algorithm is a graph analysis algorithm, which is used for finding transitive closure of a relation R and also for finding shortest paths in a weighted graph. Weighted graph may be with positive or negative edge weights (but with no negative cycles). Single execution of the algorithm finds lengths of all the shortest paths between each and every pair of vertices present in the graph ,thus it does not returns details of the path themselves. To find all pair of vertices in a graph Floyd −Warshall algorithm will be used. This algorithm is competitive for dense graphs and uses adjacency matrices as opposed to adjacency lists. Consider an instance for TSP is given by W as, Represent the directed, edge-weighted graph in adjacency-matrix form.

W = matrix of weights = $\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{pmatrix}$

.

w_{ij} is the weight of edge (i, j), or infinity if there is no such edge.

. Return a matrix D, where each entry d_{ij} is d(i,j). Could also return a predecessor matrix, P, where each entry p_{ij} is the predecessor of j on the shortest path from i. Consider intermediate vertices of a path:

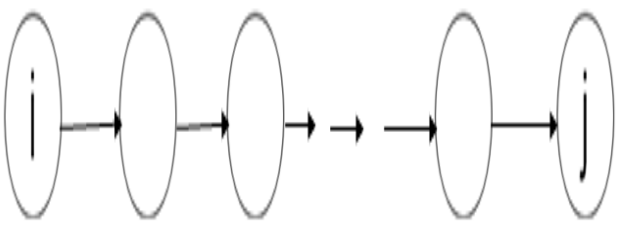

Figure 4: Intermediate vertices of path.

Say we know the length of the shortest path from i to j whose intermediate vertices are only those with numbers 1, 2, ..., k-1. Call this length Now to extend this from k-1 to k we can use

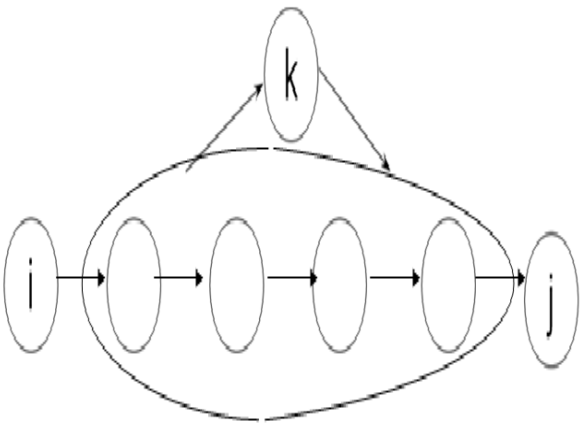

Figure 5: Alternative Intermediate vertices of path using Floyd−Warshall algorithm

Two possibilities: 1. Going through the vertex k does n t help− the path through vertices 1...k-1 is still the shortest. 2. There is a shorter path consisting of two sub paths, one from i to k and one from k to j. Each sub path passes only through vertices numbered 1 to k-1 Thus,

$d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$

Also, $d_{ij}^{(0)} = w_{ij}$

(since there are no intermediate vertices.) When k = |V|, we're done. Let n be |V|, the number of ver-tices. To find all n^2 of shortestPath(i,j,k) (for all i and j) from those of shortestPath(i,j,k−1) requires $2n^2$ op-erations. Since we begin with shortestPath(i,j,0) = edge-Cost(i,j) and compute the sequence of n matrices shortest-Path(i,j,1), shortestPath(i,j,2),..., shortestPath(i,j,n), the total number of operations used is n . $2n^2 = 2n^2$. There-fore, the complexity of the algorithm is $\theta(n^3)$.[5],[6]

III. COMPARISON BETWEEN PATH FINDING ALGORITHM

The worst case time complexities of all above algorithms are given as follows

Table 1: Comparisons between path finding algorithm

Algorithm	Time complexity	Space complexity
Dijkstra	O(|V||E|)	O(|V||E|)
Bellman-Ford	O(|V||E|)	O(|V|)
A*	O(|E|)	O(|V|)
Floyd-Warshall	$O(\|V\|^3)$	$\theta(\|V\|^2)$

IV. CONCLUSION

Having examined a variety of path finding methods, we can conclude that with the use of Floyd-Warshall algorithm in our system, the routing of bus paths can be done accurately. From the table, it is clear that the Floyd-Warshall algorithm provides acceptable time and space

requirements even in the worst case situations. It would greatly reduce the need of human thought process to plan the routes by providing optimal path in lesser time. We have considered certain assumptions in this design process. The congestion may vary in real time scenario as our system would just provide congestion based on past data, and there may be traffic jams etc. Also, while calculating the time, it's assumed that the bus won't have any stoppages in between and would head directly to the pick up or drop points. It's also assumed that the buses are always available at the time of dropping the passengers back home. The arrival time of the buses isn't considered.

ACKNOWLEDGEMENT

We feel great pleasure in submitting this literature survey paper on shortest path finding algorithms. We wish to express true sense of gratitude towards our project guide, prof. V.A.Mishra and co-guide prof. R.R.jadhav who at very discrete step in study of this paper contributed her valuable guidance and help to solve every problem that arose.

REFERENCES

[1] Fast Shortest Path Algorithms for Large Road Net-works Faramroze Engineer Department of Engineer-ing Science University of Auckland New Zealand

[2] Heuristic shortest path algorithms for transportation applications: State of the art L. Fua,*, D. Sunb, L.R. Rilettc aDepartment of Civil Engineering, Uni-versity of Waterloo, Waterloo, ON, Canada N2L 3G1 bCollege of Automation, Chongqing University, Chongqing, 400044, China cMid-America Transporta tion Center, University of Nebraska-Lincoln, W339 Nebraska Hall, P.O. Box 880531, Lincoln, NE 68588-0531, USA

[3] Mining the Shortest Path within a Travel Time Constraint in Road Network Environments Eric Hsueh Chan Lu, Chia-Ching Lin, and Vincent S. Tseng Department of Computer Science and Information Engineering National Cheng-Kung University Tainan, Taiwan, R.O.C.

[4] Design and implememntation of multiparameter Dijkstra' s(MPD) algorithm:A shortest path algorithm for real-road networks.(September 2011) 1Nishtha keswani,2 Dinesh Gopalani [1]asistant professor, central universal of Rajsthan, India
[2]assistant professor, Malviya National Institute Of Technology, Jaipur. (IJAER) 2011, Vol. No. 2, Issue No. III, September 2011

[5] Shortest path algorithms, Wikipedia, the free encyclo-pedia en.wikipedia.org/wiki/Shortest_path_problem

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein.MIT Press/McGraw-Hill, ISBN: 0-262-03293-7.I Chapter 25 − All-Pairs Shortest Paths

[7] Algorithm lectures note on shortest path Jeff Erickson http://www.cs.uiuc.edu/ jeffe/teaching/algorithms/ I Lecture 20 − All-pairs shortest paths

[8] Open Shortest Path First (OSPF) Conformance and Performance Testing

[9] A* search algorithm, Wikipedia, the free encyclo-pedia en.wikipedia.org/wiki/A*_search_algorithm