Resourceful Solution for Finding Shortest Path in Networks

K.Jyothana[#], CH.Mounica[*], K.Anudeep[#]

[#]*Dept of CSE, K.L.University*
Vaddeswaram,Guntur
jyoshkolluri@gmail.com
[1]

[3]kataragaddaanudeep@gmail.com

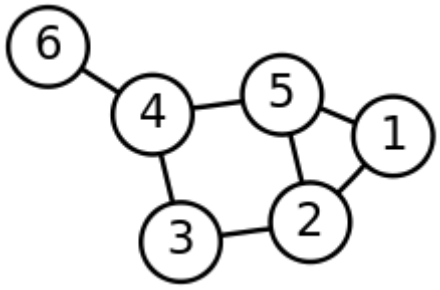
[*] *Dept of CSE, K.L.University*
Vaddeswaram,Guntur
[2] mounica@gmail.com

***Abstract*— In graph theory, the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized. An algorithm on the network shortest path problem by gradually eliminating loops on a network is put forward. Using the wagon routing arrowhead line, we start from the origin, plot the routing arrowhead lines in the current loop and the adjacent loops, and decide which edge the arrowhead pointed to should be moved; then according to the structure of the candidate edges to be removed and certain regulations, remove one edge to enlarge the current loop; select the loop nearest to the origin and repeat the above process, until obtain the shortest routing tree taking the origin as its root. The natural linear programming(LP) formulation for the shortest path problem carried out shows that the algorithm is simple, practical, knowable, and suitable for manual searching the shortest route on a simple non-directional network.**

***Keywords*— Include at least 5 keywords or phrases**

I. INTRODUCTION

In graph theory, the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized. This is analogous to the problem of finding the shortest path between two intersections on a road map: the graph's vertices correspond to intersections and the edges correspond to road segments, each weighted by the length of its road segment.



(6, 4, 5, 1) and (6, 4, 3, 2, 1) are both paths between vertices 6 and 1

.

A. Road networks

A road network can be considered as a graph with positive weights. The nodes represent road junctions and each edge of the graph is associated with a road segment between two junctions. The weight of an edge may correspond to the length of the associated road segment, the time needed to traverse the segment or the cost of traversing the segment. Using directed edges it is also possible to model one-way streets. Such graphs are special in the sense that some edges are more important than others for long distance travel (e.g. highways). This property has been formalized using the notion of highway dimension. There are a great number of algorithms that exploit this property and are therefore able to compute the shortest path a lot quicker than would be possible on general graphs.

All of these algorithms work in two phases. In the first phase, the graph is preprocessed without knowing the source or target node. This phase may take several days for realistic data and some techniques. The second phase is the query phase. In this phase, source and target node are known. The running time of the second phase is generally less than a second. The idea is that the road network is static, so the preprocessing phase can be done once and used for a large number of queries on the same road network.

II. DEFINITION

The shortest path problem can be defined for graphs whether undirected, directed, or mixed. It is defined here for undirected graphs; for directed graphs the definition of path requires that consecutive vertices be connected by an appropriate directed edge.

Two vertices are adjacent when they are both incident to a common edge. A path in an undirected graph is a sequence of vertices $P = (v_1, v_2, \ldots, v_n) \in V \times V \times \ldots \times V$ such that v_i is adjacent to v_{i+1} for $1 \leq i < n$. Such a path P is called a path of length n from v_1 to v_n. (The v_i are variables; their numbering here relates to their position in the sequence and needs not to relate to any canonical labelling of the vertices.)

Let $e_{i,j}$ be the edge incident to both v_i and v_j. Given a real-valued weight function $f : E \to \mathbb{R}$, and an undirected (simple)

graph G, the shortest path from v to v' is the path $P = (v_1, v_2, \ldots, v_n)$ (where $v_1 = v$ and $v_n = v'$) that over all possible n minimizes the sum

$$\sum_{i=1}^{n-1} f(e_{i,i+1}).$$

When each edge in the graph has unit weight or $f : E \to \{1\}$, this is equivalent to finding the path with fewest edges.

The problem is also sometimes called the single-pair shortest path problem, to distinguish it from the following variations:

1) The single-source shortest path problem, in which we have to find shortest paths from a source vertex *v* to all other vertices in the graph.
2) The single-destination shortest path problem, in which we have to find shortest paths from all vertices in the directed graph to a single destination vertex *v*. This can be reduced to the single-source shortest path problem by reversing the arcs in the directed graph.
3) The all-pairs shortest path problem, in which we have to find shortest paths between every pair of vertices *v*, *v'* in the graph.

III. Notations and Definitions

Let Graph G = (V,E), where V is the set of Vertices V={v_1, v_2, ...,V_n }, E stands for the set of Edges, E = { e1,e2,…,e3}, the non-negative real number b(e) = b(v_i , v_j) = $b_{i,j}$ denotes the weight of b(v_i , v_j), then Gis called a non-negative weighted graph, and the route with the shortest distance from node S to node t, s Є V, t Є V is called the shortest route from s to t.

A loop that has no other loops in it is called a simplest loop. Vertex and edge Shared by two simplest loops are called respectively common vertex and common edge. The loop currently studied is called the current loop, and the loops with currently edge are called the adjacent loop to each other.

In the simplest loop, the arrowhead line starting from the node nearest to the origin and pointing to the place half loop length away from the vertex is called the routing line. The vertex from which the routing line is sent is called the starting point on the current loop is called the current starting point. If the routing line points to the adjacent loop the edge is located in the current loop is called the main adjacent loop; the other adjacent loops are called the subroutine loops; the edge to be pointed to by the routing line in the current loop is called main candidate edge to be eliminated; if the routing line of the subroutine loop points to the current loop, the shared edge is called subroutine edge of the current loop; if the routing line of a loop points to a vertex of it, the vertex is called the equal-split point of the loop, its adjacent edges are called candidate edges to be eliminated. If there is more than one of the routes that have the same length, they have the same length, they called the parallel routes.

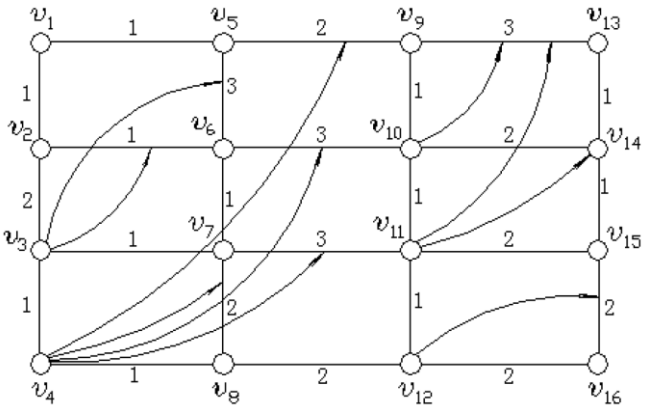


Fig. 1: Network Graph with weights

In figure 1, (v_1, v_5 ,v_6 ,v_7, v_{11} ,v_{12} ,v_8, v_4 ,v_3,v_2, v_1) is a loop; (v_3, v_7 ,v_8 ,v_4 ,v_3) is a simplest loop. (v_7, v_8) is a common edge of loop (v_3, v_7 ,v_8 ,v_4 ,v_3) and loop (v_7, v_{11} ,v_{12} v_8, v_7); vertex v_7 and v_8 are the common vertices of the two loops; (v_3, v_7 ,v_8 ,v_4 ,v_3) and (v_7, v_{11} ,v_{12} v_8, v_7) are the adjacent loops to each other. In loop (v_3, v_7 ,v_8 ,v_4 ,v_3), the arrowhead line issued from v_4 is the routing line of the loop, v_4 is the start point. The routing line points to common edge (v_7 ,v_8), also means to point the adjacent loop (v_7, v_{11} ,v_{12} v_8, v_7). If loop (v_3, v_7 ,v_8 ,v_4 ,v_3) is the current loop, then edge (v_7 ,v_8) is its main edge, loop (v_7, v_{11} ,v_{12} ,v_8, v_7) is the main adjacent loop, loop (v_2 ,v_6 ,v_7,v_3,v_2) is its subordinate adjacent loop. In loop (v_{10} ,v_{14} ,v_{15}, v_{11} ,v_{10}), the routing line points to vertex v_{14}, so is the equal-split point of the loop; edge (v_{10},v_{14}) and (v_{14} ,v_{15}) is the two candidate adjacent edge of v_{14} form being eliminated.

IV. Algorithm

In simplest loop, the routing line divides the loop into two equal-length half-loop. Therefore, the route from the starting point to any vertex of the loop should pass through the half-loop that the vertex itself is located in. For example, in figure 1, the routing from v_4 to v_7 should pass through (v_4 ,v_7 ,v_8).

If the routing line points to any vertex in the main adjacent loop will not pass through the common edge. In figure 1, in loop(v_3, v_7 ,v_8 ,v_4 ,v_3), because the routing line issuing from v_4 points to the main edge (v_7 ,v_8), the routing from v_4 to v_{12} via (v_4 , v_8 ,v_{12}) obviously is shorter than the one via (v_4 ,v_3 ,v_7, v_8 ,v_{12}).

In figure 2, v_3 is the starting point; edge (v_4 ,v_7) is the main edge in loop (v_2, v_5 ,v_4 ,v_7, v_8 ,v_9 ,v_6,v_3,v_2); (v_4,v_5) is the subroutine edge. If we only consider the routing in current loop (v_2, v_5 ,v_4 ,v_7, v_8 ,v_9 ,v_6,v_3,v_2), the shortest route from v_3 to v_7 should be via v_6, v_9 ,v_8 , but though careful observation we will find there are two routes from v_2 to v_4 and the routing line issuing v_2 from points to edge (v_4 ,v_5) which means that the shortest route form v_2 to v_4 should pass through not edge (v_4,v_5) but v_1.Therefore, the edge (v_4, v_5) should be removed. In the newly formed current loop (v_1 ,v_4 ,v_7, v_8 ,v_9 ,v_6,v_3,v_2,v_1), the routing line issuing from v_3 points to (v_7, v_8). So the route from v_3 to v_7 apparently should pass through vertices v_2, v_1 ,v_4 instead v_6,v_9,v_8.

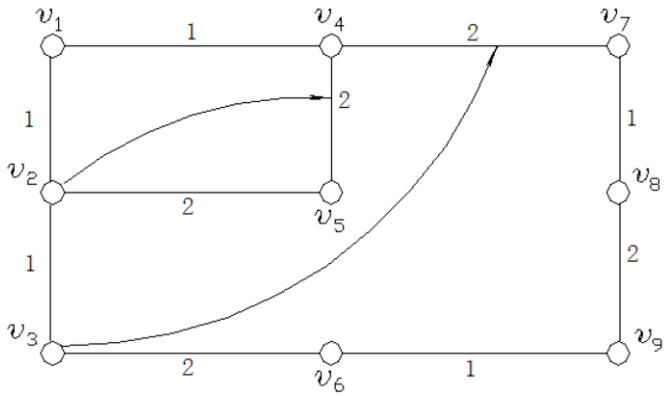


Fig. 2: Effect to Shortest route by Subordinate Edge

Steps of the algorithm:

Step 1: Let the origin vertex for the starting point and the loop that starting point is located in for the current loop;

Step 2: If there is no routing line issuing from the starting point, draw it;

Step 3: Check each of the subordinate adjacent loops to find out that if there is a routing line in it; if not draw a routing line from the vertex nearest to the origin;

Step 4: If in the current loop there exists a subroutine edge, eliminate the longest edge among the adjacent candidate edges to be eliminated;

Step 5: If all the loops have been broken, the result has been got which means the calculated process has come to its end; otherwise, continue to the next;

Step 6: If the edge to be removed is the common edge, combine the current loop to from the new current loop, and still take the current starting point, then go to Step 2; otherwise, if the removed edge is not common edge, find a vertex which is nearest to the origin and is the loop, take this vertex as the starting point and the loop that the starting point is located in as the current loop, turn to step 2.

In case of parallel route, the candidate adjacent edge having been removed should be added to the shortest path network whose root is the origin.

V. Linear programming formulation

here is a natural linear programming formulation for the shortest path problem, given below. It is very simple compared to most other uses of linear programs in discrete optimization, however it illustrates connections to other concepts.

Given a directed graph (*V*, *A*) with source node *s*, target node *t*, and cost w_{ij} for each arc (*i*, *j*) in *A*, consider the program with variables x_{ij} minimize

$$\sum_{ij \in A} w_{ij} x_{ij}$$

subject to $x \geq 0$ and for all *i*,

$$\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 1, & \text{if } i = s; \\ -1, & \text{if } i = t; \\ 0, & \text{otherwise.} \end{cases}$$

This LP, which is common fodder for operations research courses, has the special property that it is integral; more specifically, every basic optimal solution (when one exists) has all variables equal to 0 or 1, and the set of edges whose variables equal 1 form an *s*-*t* dipath. See Ahuja et al. for one proof, although the origin of this approach dates back to mid-20th century.

The dual for this linear program is maximize $y_t - y_s$ subject to for all *ij*, $y_j - y_i \leq w_{ij}$ and feasible duals correspond to the concept of a consistent heuristic for the A* algorithm for shortest paths. For any feasible dual *y* the reduced costs $w'_{ij} = w_{ij} - y_j + y_i$ are nonnegative and A* essentially runs Dijkstra's algorithm on these reduced costs.

VI. Applications

Shortest path algorithms are applied to automatically find directions between physical locations, such as driving directions on web mapping websites like Mapquest or Google Maps. For this application fast specialized algorithms are available.

If one represents a nondeterministic abstract machine as a graph where vertices describe states and edges describe possible transitions, shortest path algorithms can be used to find an optimal sequence of choices to reach a certain goal state, or to establish lower bounds on the time needed to reach a given state. For example, if vertices represents the states of a puzzle like a Rubik's Cube and each directed edge corresponds to a single move or turn, shortest path algorithms can be used to find a solution that uses the minimum possible number of moves.

In a networking or telecommunications mindset, this shortest path problem is sometimes called the min-delay path problem and usually tied with a widest path problem. For example, the algorithm may seek the shortest (min-delay) widest path, or widest shortest (min-delay) path.

A more lighthearted application is the games of "six degrees of separation" that try to find the shortest path in graphs like movie stars appearing in the same film.

Other applications, often studied in operations research, include plant and facility layout, robotics, transportation, and VLSI design".

VII. Conclusions

An algorithm on the network shortest path problem by gradually eliminating loops on a network is put forward. Taking routing arrowhead line as a tool, we remove some of the edges in the giving non-directional network according to the certain regulations to eliminate the loops. At the beginning, the edges in the network are relatively dense. With the algorithm progresses, the number of edges on the network gradually reduced and eventually the number of edges is equal to the number of vertices min us 1, to from the shortest routing tree with the starting point as its root. Finally, the candidate adjacent edges to be eliminated that function as the parallel routes and have been removed should be added to the shortest routing tree. The natural linear programming(LP) formulation for the shortest path problem optimised results. gives The

algorithm mentioned above is simple, practical and easy to master, and the calculation process is intuitive, so it adapts to solve the shortest routing problem on non-directional network by hand, especially to driving routing, pipe network planning, and other network shortest path problem.

REFERENCES

[1] Xie Jin-bao, "Simplified Algorithm on Network Shortest Path Problem", IEEE Conference 2011.

[2] Hart P E and Nilson N J. (1968) A formal basis of the heuristic determination of minimum cost paths. *IEEE Transactions of Systems Science and Cybernetics* 4(2), 100-107.

[3] Pearsons J. (1998) *Heuristic Search in Route Finding*. Master's Thesis, University of Auckland.

[4] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin (1993). *Network Flows: Theory, Algorithms and Applications*. Prentice Hall. ISBN 0-13-617549-X.

[5] Leyzorek, M.; Gray, R. S.; Johnson, A. A.; Ladew, W. C.; Meaker, S. R., Jr.; Petry, R. M.; Seitz, R. N. (1957). *Investigation of Model Techniques — First Annual Report — 6 June 1956 — 1 July 1957 — A Study of Model Techniques for Communication Systems*. Cleveland, Ohio: Case Institute of Technology.

[6] Moore, E. F. (1959). "The shortest path through a maze". *Proceedings of an International Symposium on the Theory of Switching (Cambridge, Massachusetts, 2–5 April 1957)*. Cambridge: Harvard University Press. pp. 285–292.

