ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 1, January 2013

Survey on Certain Algorithms Computing Best Possible Routes for Transportation Enquiry Services

***Neha Choubey*[1]**
M. Tech. Scholar, Department of Computer Science and Engg. CSIT, Durg (CG) INDIA

Mr. Bhupesh Kr. Gupta[2]
Assistant Professor, Department of Computer Science and Engg. CSIT, Durg (CG) INDIA

Abstract- Shortest Path problems are very common in road network applications where the optimal routings have to be found. As the traffic condition among a city changes from time to time and there are usually a huge amounts of problems occur at any moment, it needs to quickly find the solution. Therefore, the efficiency of the algorithm is very important. Some approaches take advantage of preprocessing that compute results before demanding. These results are saved in memory and could be used directly when a new request comes up. This can be inapplicable if the devices have limited memory and external storage. This project aims only at the single source shortest path problems and intends to obtain some general conclusions by examining these approaches.

IndexTerms*—Shortestpath,Roadnetworks,Dijkstra,Bell manford,Prims,A

I. INTRODUCTION

The shortest path problem is one of the fundamental problems with numerous applications. In this paper we study most common variants of the problem, where the goal is to find a point-to-point shortest path in a directed graph. Our goal is to find a fast algorithm for answering point-to-point shortest path queries. Due to the nature of routing applications, we need flexible and efficient shortest path procedures, both from a processing time point of view and also in terms of the memory requirements. Prior research does not provide a clear direction for choosing an algorithm when one faces the problem of computing shortest paths on real road networks.

In this paper we are going through a case study of four algorithms to find the shortest path between two places. In this paper we survey a number of recent algorithmic techniques for maintaining shortest routes in dynamic graphs. We focus on the all-pairs version of the problem, where one is interested in maintaining information about shortest paths between each pair of nodes. Many of the techniques described in this paper can be specialized for the case where only shortest paths between a subset of nodes in the network have to be maintained.

A. *The shortest path problem*

A road traffic network is represented by a digraph $G(N, A)$ that consists of a set of nodes N and a set of arcs A (or links used in this paper). Denote the number of nodes $|N|=n$ and the number of links $|A|=m$. A link $a = (i, j) \in A$ is directed from node i to node j and has an associated generalized cost cij. The generalized cost represents the impedance of an individual vehicle going through that link and is usually described by link travel time, link length, tolls, etc. Without losing generality, the term link travel time is used mostly in this paper. A path from an origin (o) to destination (d) may be defined as a sequential list of links: $(o, j), (i, d)$ and the travel time of the path is the sum of travel times on the individual links [7] [13].

The problem is to find the path that has the minimum total travel time from the origin node to the destination node. The

All Rights Reserved © 2013 IJARCET

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.

B .*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 (i.e. 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 [7].

II. SOME SHORTEST PATH ALGORITHMS

We have gone through study of various algorithms for the computation of shortest path some of them are:

A. DIJKSTRA ALGORITHM:

Dijkstra's algorithm, given by Dutch computer scientist Edsger Dijkstra in 1959, is a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge path costs, producing a shortest path tree. This algorithm is often used in network routing protocols [4] [5].

If all edge weights are non negative, all shortest-path weights must exist.Dijkstra's algorithm is called the single-source shortest path. It is also known as the single source shortest path problem. It computes length of the shortest path from the source to each of the remaining vertices in the graph. The single source shortest path problem can be described as follows:

Let G= {V, E} be a directed weighted graph with V having the set of vertices. The special vertex s in V, where s is the source and let for any edge e in E, Edge Cost(e) be the length of edge e. All the weights in the graph should be positive.

Directed graph can be defined as an ordered pair G: = (V,E) with V is a set, whose elements are called vertices or nodes and E is a set of ordered pairs of vertices, called directed edges, arcs, or arrows. Directed graphs are also known as digraph [4] [9].

Directed-weighted graph is a directed graph with weight attached to each of the edge of the graph.

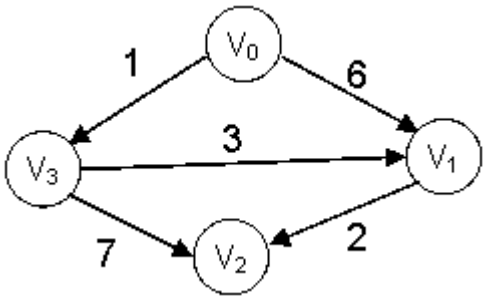

Fig: 1 Directed weighted graph

Dijkstra's –A Greedy Algorithm

Greedy algorithms use problem solving methods based on actions to see if there's a better long term strategy. Dijkstra's algorithm uses the greedy approach to solve the single source shortest problem. It repeatedly selects from the unselected vertices, vertex v nearest to source s and declares the distance to be the actual shortest distance from s to v. The edges of v are then checked to see if their destination can be reached by v followed by the relevant outgoing edges [5] [9].

Basic Idea of Algorithm: Greedy.

1. Maintain a set *S* of vertices whose shortest path distance from *s* is known.
2. At each step add to *S* the vertex *v* in (*V* - *S*) whose distance estimate from *s* is minimal.
3. Update the distance estimates of vertices adjacent to *v*.

For a given source vertex in the graph, the algorithm finds the path with the lowest cost between that vertex and every other vertex. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been found.

Application: For example, if the vertices of the graph represent cities and edge path costs represent driving

distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities. As a result, the shortest path first is widely used in network routing protocols.

B. Bellman ford Algorithm:

The Bellman-Ford algorithms compute single-source shortest paths in a weighted diagraph. The algorithm is named after its developers, Richard Bellman and Lester Ford, Jr. For graphs with non-negative weights, Dijkstra's algorithm solves the problem. The Bellman-Ford algorithm is used primarily for graphs with negative weights. The algorithm can detect negative cycles and report their existence, but it cannot produce a correct "shortest path" if a negative cycle is reachable from the source [5] [6].

Basic Idea of Algorithm:

1. Uses Relaxation algorithm
2. Relaxations of edges in smart way
3. Label edges $e_1, e_2, ..., e_m$

Relax in this order: $e_1, e_2, ..., e_m$; $e_1, e_2, ..., e_m$; ...; $e_1, e_2, ..., e_m$; for $|V|$-1 repetitions.

It uses the same concept as that of Dijkstra's algorithm but can handle negative edges as well. It has a better running time than that of Dijkstra's algorithm. For graphs with only non-negative edge weights, the faster than Dijkstra algorithm also solves the problem. However, if a graph contains a "negative cycle", i.e., a cycle whose edges sum to a negative value, then walks of arbitrarily low weight can be constructed by repeatedly following the cycle, so there may not be a *shortest* path. In such a case, the Bellman-Ford algorithm can detect negative cycles and report their existence, but it cannot produce a correct "shortest path" answer if a negative cycle is reachable from the source[5][6].

Basic Idea of Algorithm: Dynamic Programming Approach

Dynamic Programming given by mathematician Richard Bellman in 1953 solves problems efficiently by overlapping sub problems and optimal substructure.

Dynamic Programming strategy logic is

1. Break the problem into smaller sub problems.
2. Solve these problems optimally using this three-step process recursively.
3. Use these optimal solutions to construct an optimal solution for the original problem.

The sub problem search leveling has no limitations till found either the "atom" sub problem or a sub problem easy to solve. By overlapping sub problems we mean reuse of partial computations performed over low level sub problems. It saves time but as a counterpart it takes too much memory. Another characteristic of Dynamic Programming is "memoization" that stands for keeping any computation potentially apt for ulterior reuse. Memoization requires high skill programming.

Finding negative cycles

Negative cycles is a problem, when we have to find shortest path, it prevents the algorithm from finding a correct answer. After finding a negative cycle it terminates, so this algorithm can be used for applications in which this is the target to be sought - for example in cycle-canceling techniques in network flow analysis**.**

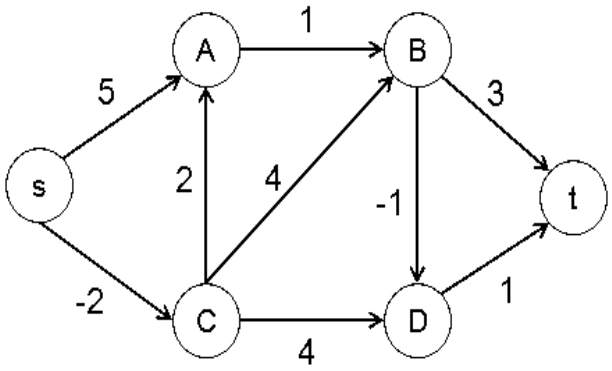

Fig: 2 Graph with negative edge

The algorithm simply relaxes all the edges, and does these |V|-1 times, where $|V|$ is the number of vertices in the graph. The repetitions allow minimum distances to propagate accurately throughout the graph, since in the absence of negative cycles, the shortest path can visit each node at most only once [5] [6].

Applications in routing

A distributed variant of the Bellman–Ford algorithm is used in distance-vector routing protocols, for example the Routing Information Protocol (RIP). We can say that the algorithm is distributed because it involves a number of nodes (routers) within an Autonomous system, a collection of IP networks typically owned by an ISP.

All Rights Reserved © 2013 IJARCET

C. Prims Algorithm:

Prim's algorithm is an MST algorithm that works much like Dijkstra's algorithm does for shortest path trees. Prim's algorithm finds a minimum spanning tree for a connected weighted graph. It implies that it finds a subset of edges that form a tree where the total weight of all the edges in the tree is minimized. It is sometimes called the DJP algorithm or Jarnik algorithm Prim's algorithm is an MST algorithm that works much like Dijkstra's algorithm does for shortest path trees [6].

The reason is that there already was one path between the endpoints (since it's a spanning tree), and now there are two. If you then remove any edge in the cycle, you get back a spanning tree (removing one edge from a cycle cannot disconnect a graph).

Running time:

We can implement this in the same was as Dijkstra's algorithm, getting an O(mlog n) running time if we use a standard heap, or O(m + n log n) running time if we use a Fibonacci heap. The only difference with Dijkstra's algorithm is that when we store the neighbors of T in a heap, we use priority values equal to the shortest edge connecting them to T (rather than the smallest sum of "edge length plus distance of endpoint to s")[6].

Basic Idea of Algorithm

1. This algorithm repeatedly chooses the smallest-weight edge from the tree so far to the other vertices.
2. If a spanning tree has a weightier edge between VT and V − VT, it can be improved by replacing it with e.
3. We can put VT edges into a priority queue, then dequeue edges until one goes between VT and V − VT.
4. Prim's Algorithm using a binary heap to implement a priority queue is O (E logE) = O (E log V)

.

D..*A* algorithm*:

The A* algorithm by Hart and Nilsson formalized the concept of integrating a heuristic into a search procedure.

It is a graph/tree search algorithm that finds a path from a given initial node to a given goal node It employs a "heuristic estimate" h(x) that gives an estimate of the best route that goes through that node. It visits the nodes in order of this heuristic estimate. It follows the approach of best first search [4] [9] [13].

However the physical nature of real road networks motivates investigation into the possible use of heuristic solutions that exploit the near-Euclidean network structure to reduce solution times while hopefully obtaining near optimal paths. For most of these heuristics the goal is to bias a more focused search towards the destination. As we shall see, incorporating heuristic knowledge into a search can dramatically reduce solution times [4] [9].

So far we have examined search techniques that can be generalized for any network (as long as it does not contain negative length cycles). However the physical nature of real road networks motivates investigation into the possible use of heuristic solutions that exploit the near-Euclidean network structure to reduce solution times while hopefully obtaining near optimal paths. For most of these heuristics the goal is to bias a more focused search towards the destination. As we shall see, incorporating heuristic knowledge into a search can dramatically reduce solution times [9].

Instead of choosing the next node to label permanently as that with the least cost (as measured from the start node), the choice of node is based on the cost from the start node plus an estimate of proximity to the destination (a heuristic estimate). To build a shortest path from the origin *s* to the destination *t*, we use the original distance from *s* accumulated along the edges (as in Dijkstra's algorithm) plus an estimate of the distance to *t*. Thus we use global information about our network to guide the search for the shortest path from *s* to *t*. This algorithm places more importance on paths leading towards *t* than paths moving away from *t* [4] [9] [13].

In essence the A* algorithm combines two pieces of information:

1. The current knowledge available about the upper bounds (given by the distance labels *d (i)*), and
2. An estimate of the distance from a leaf node of the search tree to the destination.

There are several ways to estimate the lower bound from a leaf node in the search tree to the destination node. These estimations are carried out by so called "evaluation" functions. The closer this estimate is to a tight lower bound on the distance to the destination, the better the quality of the

A* Search. Hence the merits of an A* search depends highly on the evaluation function *h (i, j) [2] [9]*.

III CONCLUSION

Going through comparative study of various algorithms we came through various drawbacks of algorithm and also advantages of these approaches. Each and every algorithm is unique and plays specific role in obtaining best possible paths for the users. As per study we came to know that Dijkstra algorithm consumes lot of time and doesn't work well on dense networks. Prims algorithm cannot be applied for large graphs. A* algorithm based on heuristics works well and fast for large graphs. It depends on the network and its complexicity and also time dependency that to use the algorithm that best suits the networks. Still researches have been going on for fastest algorithms for road networks to achieve best possible route and the optimal path.

REFERENCES

[1] Andrew V. Goldberg, Chris Harrelson," Computing the Shortest Path: A_ Search Meets Graph Theory" Technical Report MSR-TR-2004-24, March 2003

[2] Andrew V. Goldberg1, Chris Harrelson," Computing the Shortest Path: A* Search Meets Graph Theory" Technical Report MSR-TR-2004-24

[3] Camil Demetrescu, Giuseppe F. Italia no," Algorithmic Techniques for Maintaining Shortest Routes in Dynamic Networks", Dipartimento di Informatica e Sistemistica Universit`a di Roma "La Sapienza" Roma, Italy, WCAN 2006

[4] Liang Dai," Fast Shortest Path Algorithm for Road Network and Implementation", Carleton University School of Computer Science COMP 4905 HONOURS PROJECT
Fall Term, 2005.

[5] Faramroze Engineer," Fast Shortest Path Algorithms for Large Road Networks", Department of Engineering Science University of Auckland
New Zealand

[6] Luciana S. Buriol," Speeding up Dynamic Shortest-Path Algorithms", *INFORMS Journal on Computing* Vol. 20, No. 2, spring 2008.

[7] L. Fua, D. Sunb, L.R. Rilett," Heuristic shortest path algorithms for transportation applications: State of the art", Computers & Operations Research 33 (2006) 3324–3343

[8] Subadra; M.Bhagavan Das; and C.Rama Seshagiri Rao," Directed Graph Algorithms for Tours – A Case Study", Scholar link Research Institute Journals, (ISSN: 2141-7016), 2011.

[9] Dmitry S. Yershov and Steven M. LaValle," Simplicial Dijkstra and A* Algorithms: From Graphs to Continuous Spaces" Department of Computer Science, University of Illinois at Urbana-Champaign, 201 North Goodwin Ave., Urbana, IL 61801, USA,August 2012.

[10] John *Pucher and Nisha Korattyswaroopam,"* The Crisis of Public Transport in India: Overwhelming Needs but Limited Resources," *Journal of Public Transportation, Vol. 7, No. 4, 2004*

[11] Donghui Zhang," Fastest-Path Computation", College of Computer & Information Science North-eastern University

[12] Martin Holzer, Frank Schulz, Dorothea Wagner, Thomas Willhalm," Combining Speed-up Techniques for Shortest-Path Computations" March 9, 2006

[13] L. Fua, D. Sunb, L.R. Rilettc,"Heuristic shortest path algorithms for transportation applications: State of the art", Computers & Operations Research 33 (2006) 3324–3343

[14]Anthony J. Richardson, Elizabeth S. Ampt, Arnim H. Meyburg, "Survey Methods for Transport Planning"

Authors

[1Neha Choubey

She , received her B.E. (Computer Sc.) in year 2010 and pursuing for M.Tech. (Computer Sc.) from Chhatrapati Shivaji Institute of Technology (CSIT), Durg, Chhattisgarh, India. Her interests are Computer Network, Operating Systems and Network Security.

[2] Mr. Bhupesh Kr. Gupta

He received his B.Tech UPTU (LKO) in year 2004 and M.Tech. (Eng. Systems.) from Dayalbagh Educational Institute Dayalbagh Agra, India. Presently Assistant Professor (Computer Sc.) Dept. Chhatrapati Shivaji Institute of Technology (CSIT) ,Durg, (C.G), India. His interests are Mobile And Wireless Computing,, Optimization-Techniques and Operating System.

All Rights Reserved © 2013 IJARCET