AN ESTIMATE OF HEURISTIC APPROACH TO OBTAIN THE SHORTEST PATH FOR REAL ROAD PUBLIC TRANSPORT NETWORK

Guruprasad Nagaraj[1] and Y S Kumaraswamy[2]

[1]Research Scholar - Dept. of Computer Science, Dr. MGR Deemed University, Chennai
[2]Professor & Head, Dept of Master of Computer Applications, Dayanand Sagar College of Engineering, Bangalore, INDIA
ngp_jyothi@yahoo.co.in, yskldswamy@yahoo.co.in

ABSTRACT

Transit systems are public transport systems which moves a large number of passengers to their destinations. Traffic congestion is becoming a serious problem in more and more modern cities; however life would come to a standstill without a proper and effectual transportation system. Finding the optimal transfer locations is essential for providing valuable passenger information in order to support journey planning. Furthermore, using this information makes possible to accelerate general-purpose shortest path algorithms, making transit assignments more efficient. However, determining optimal transfer opportunities in a dense network is not an easy task. Shortest path problems are among the most studied network flow optimization problems, with interesting applications in a range of fields. Route finding is a shortest path problem. This paper proposes a heuristic approach (minimum tree) which basically computes the impedance between the source and the destination to compute the shortest path.

Keywords

Public transport, shortest path, transit points, heuristic, impedance

1. Introduction:

Earlier transportation was not a major problem because cities were small, and all points within the city were accessible on foot within a reasonable period of time. Only the rich enjoyed the luxury of riding. With the industrial revolution of the nineteenth century, cities began to grow and expand in population and with this growth came the need for traveling greater distances, which in turn increased the demand for improved means of transportation. The current fast pace of life around the globe demands an effective, fast and reliable Public Transportation System. In a metropolis with a complicated transport network, people often do not know how to reach their destination except where they often visit. In addition, people may want to plan for the fastest or the most economical method to their destination. Due to the nature of the routing applications, we need flexible and efficient shortest path procedures, both from processing point of view and also in terms of the memory requirements. Past research in testing different shortest path algorithms suggests that Dijkstra's implementation is the best algorithm. However most of them have concentrated their focus on algorithms that guarantee optimality and have worked on tuning data structures used in implementing these algorithms.
Since no best algorithm currently exists for every kind of transportation problem, research in this field has recently moved to the design and implementation of "heuristic" shortest path procedures, which are able to capture the peculiarities of the problem under consideration and improve the run time performance of the search.

To test this approach we used parts of Bangalore road network covering seven bus routes with 23 nodes.

2. Search algorithms

One possible approach for solving shortest path problems is to precalculate and store the shortest path from every node to every possible other node, which basically allows us to answer shortest path query

in constant time. Unfortunately the required storage size and computation time grows with the square of the number of nodes. With realistic road networks in mind this processing would take years if not decades and be virtually impossible to store. Hence to overcome this problem we require real time search techniques.

3. Dijkstra's naive implementation

Dijkstra's labeling method is a central procedure in most shortest path algorithms. The output of the labeling method is an out tree from source node s to a set of nodes L. An out tree is basically a tree originating from the source node to other nodes to which the shortest distance from the source node is known. This out tree is constructed iteratively and the shortest path from s to any destination node t in the tree is obtained upon the termination of the algorithm. The information required for each node i in the labeling method while constructing the tree are as follows:

1) Distance label d(i)
2) Parent node / predecessor p(i)
3) Set of permanently labeled nodes L

4. Study Area Description and Data used

The study area (routes) taken for this work involved seven bus routes; all starting from Kempegowda Bus Station (KBS) / Majestic in the garden city of Bangalore except one which starts from City railway Station. Table 1 below gives the bus routes that was a part of the field work for the paper.

Sl. #	Bus (Route) #	Source	Destination	Distance (km)	Total # of schedules	Total # of trips
1	2	KBS	J P Nagar 6th phase	12.5	8	124
2	12	KBS	Banashankari	10.9	8	140
3	12 B	KBS	Padmanabhnagar	13.9	7	140
4	18	KBS	Jaynagar 9th block	11	5	95
5	25 *	KBS	Kuvempu Nagar (BTM Layout)	13.6	4	60
6	25 B	KBS	Kuvempu Nagar (BTM Layout)	16.3	3	48
7	25 E	KBS	J P Nagar 3rd phase	12.6	3	48

***: This route starts from city railway station.**

Not every one of us may know almost all the bus routes that are available, but if a necessity arises to travel to a new place, it is customary that someone in the boarding place comes to the rescue or the guide map guides us to the destination. Hence it augurs well for the development of a Public Transport Information System, which would provide the following.

1. Timings/frequency of the various buses.
2. Buses connecting various places of interest to the commuters, from their place of boarding,
3. Buses connecting places of tourist interest from the commuters' boarding place, and
4. Shortest possible route for the commuter with respect to time and/or distance.

5. Methodology

The study "routes" for the work was taken from KBS terminal to seven different locations in South Bangalore. Basically this terminal acted as nodal terminal (point) and the work involved the development of the algorithm to obtain the shortest or optimal route of travel for the commuter.

5.1 Fieldwork

The data required by an information system has to be inexhaustible but with its own limitations. Hence, the data collected for this purpose was extensive. First of all, the various bus stops on each of the seven routes were identified. This was done, by traveling along each of the seven routes. Both the to and fro routes had to be traversed as the bus routes in both the cases differed due to one way roads. Other information collected were the one or two nature of the road, important landmarks for the bus stops and the total time for the journey (the latter may vary depending on the hour of the day).

5.2 Algorithm

Optimal Routing is the process of delineating the best route to get from one location to another. The "best route" could be the shortest, the quickest or the most aesthetic, depending on the user's preference for defining "best". In our case, we define the "best" route, which takes the shortest path with minimum number of transit points.

The algorithm considers the following parameters

1. Time of travel from place of boarding to destination, that is, time as an impedance
2. One or Two way nature of the road
3. Minimum number of transit stops – limited to one
4. Frequency of the buses

The algorithm proceeds as follows

1) Initialize the path impedance of the tree table at zero for the node of origin and a very large number (probably infinity) for all other nodes. This large number ensures that the first encountered actual path to a node will be chosen.

2) Enter into the lists the links(i,j) that emanate directly from any node i just added to the tree.

3) For each node j included in the list, add the impedance of link (i,j) to the tree table's current total impedance to node i . This quantity represents the total impedance to node j via node i . If this value is smaller than the current tree table entry for node j, replace the current total impedance to j with the new total impedance and enter node i as the node that immediately precedes j. This operation replaces the longer path to node j with the shorter one just encountered. If the new total impedance is greater than the current tree table entry, proceed to the next link in the list.

4) Return to step 2, unless the list is empty, in which case the tree table contains the solution.

6. Implementation

The information obtained was well articulated in the form of a simple network which consists of five zonal points (i.e nodes 1 to 5), six centroidal connectors, nine street intersections (i.e nodes 6 to 14), and 13 arterial street links.

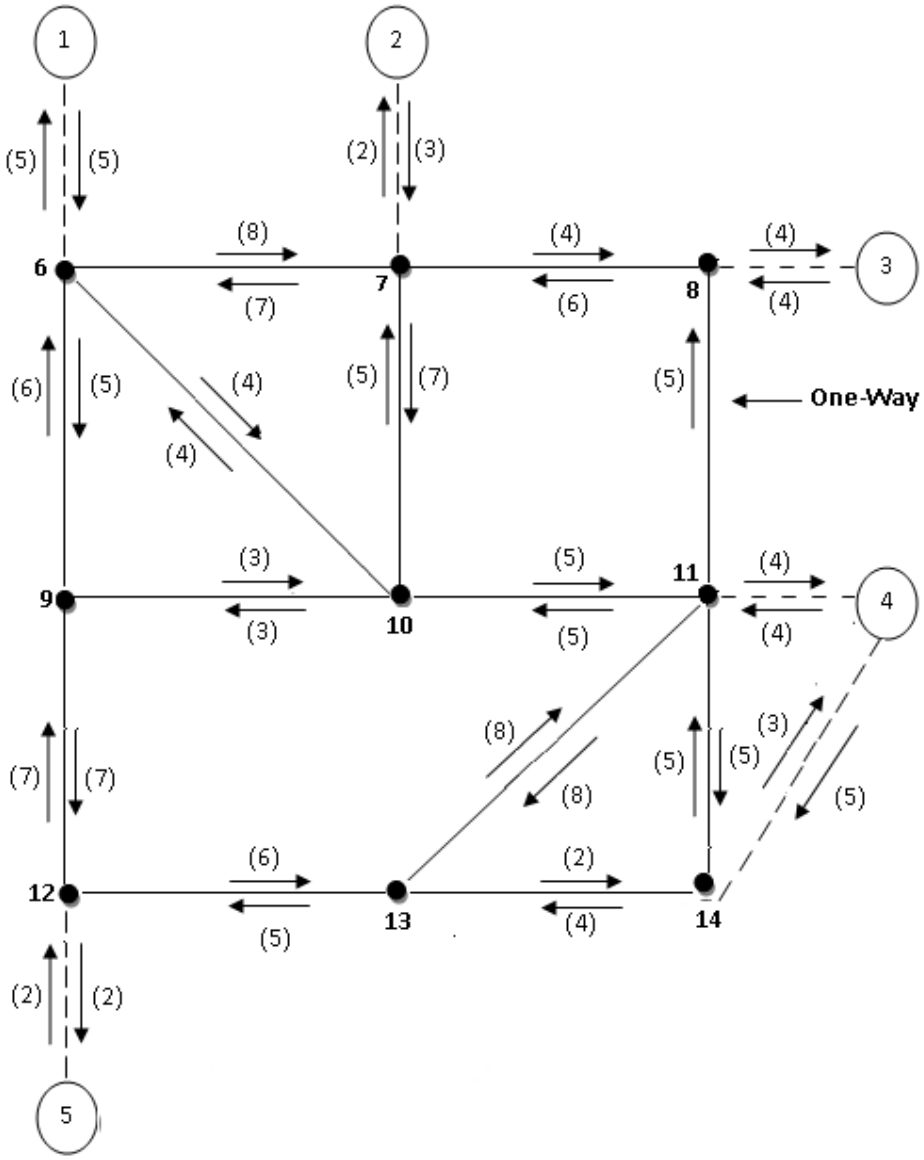


Figure 1 Network

The numerical value shown in the parentheses correspond to the link impedances in the direction shown. This network is described with the help of the link array (Table 2), each cell of which represents a possible direct link between row and column nodes.

i / j	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1						5								
2							3							
3								4						
4											4			5
5												3		
6	5						8		5	4				
7		2				7		4		7				

8			4				6							
9						6				3		7		
10						4	5		3		5			
11				4				5		5			8	5
12					2				7				6	
13											8	5		2
14				3							5		4	

Table 2 : Link array

The basic minimum tree algorithm begins at node of origin and proceeds outward, successively eliminating links that clearly do not belong on any minimum path emanating from the origin. The minimum tree can be described numerically by a tree table as shown in table 3 which contains all network nodes j including the origin, total impedance of minimum path from origin to each node and finally specifies the node i that immediately precedes node j on the minimum path.

Node	Total impedance to node j							Node preceding j						
J	I	II	III	IV	V	VI		I	II	III	IV	V	VI	
1	0							-						
2	∞			15							7			N
3	∞				21							8		O
4	∞				18							11		
5	∞				19							12		C
6	∞	5							1					H
7	∞		13							6				A
8	∞			17							7			N
9	∞		10							6				G
10	∞		9							6				E
11	∞			14							7			
12	∞			17							7			
13	∞				22							11		
14	∞				19							11		

Table 3 : Tree Table changes at the end of each stage

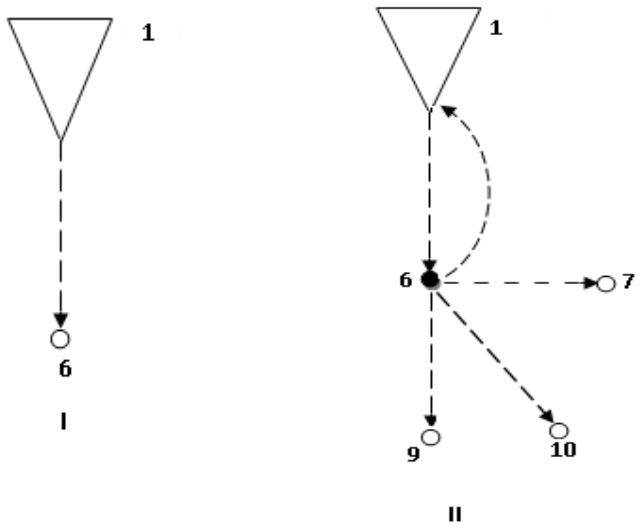
The initial condition (stage I) contains only the node of origin. All links emanating from node 1 are next entered in the list; depicted below in table 4 and are also shown by dashed lines in the final tree. (Figure 2)

Stage N	Links		Compute new path impedance	Compare to tree table stage N-1	Decision
	i	j			
II	1	6	0 + 5 = 5	5 < ∞	Accept
III	6	7	5 + 8 = 13	13 < ∞	Accept
		9	5 + 5 = 10	10 < ∞	Accept
		10	5 + 4 = 9	9 < ∞	Accept
IV	7	2	13 + 2 = 15	15 < ∞	Accept
		8	13 + 4 = 17	17 < ∞	Accept
		10	13 + 7 = 20	Reject	Reject
	9	10	10 + 3 = 13	13 > 9	Reject

		12	10 + 7 = 17	17 < ∞	Accept
	10	7	9 + 5 = 14	14 > 13	Reject
		9	9 + 3 = 12	12 > 10	Reject
		11	9 + 5 = 14	14 < ∞	Accept
V	8	3	17 + 4 = 21	21 < ∞	Accept
	11	4	14 + 4 = 18	18 < ∞	Accept
		8	14 + 5 = 19	19 > 17	Reject
		13	14 + 8 = 22	22 < ∞	Accept
		14	14 + 5 = 19	19 < ∞	Accept
	12	5	17 + 2 = 19	19 < ∞	Accept
		13	17 + 6 = 23	Reject	Reject
VI	All links emanating from nodes 3, 4, 5, 13 and 14 are rejected; the list is now empty and the algorithm terminates.				

Table 4 : List changes and related calculations

The link and the link impedances are found in row 1 of the link array (table 3). In this case there is only one entry, link 1-6 with link impedance of 5 units. The calculation of stage II are shown in table 4. The impedance of the new path is computed by adding the impedance of link 91,6) to the current tree table entry for node i=1 (i.e 0+5=5) this value is compared to the current tree table entry for node j=6 (i.e infinity). Since this path is shorter than the current value, this path is accepted. Stage II of the tree table and the tree diagram reflect this modification. All links emanating from this newly added node (node 6) are placed in the list to be considered at next stage. They are also shown by dashed lines on the partial tree diagram. At the end of stage III nodes 7, 9 and 10 are added to the tree and we proceed similarly.



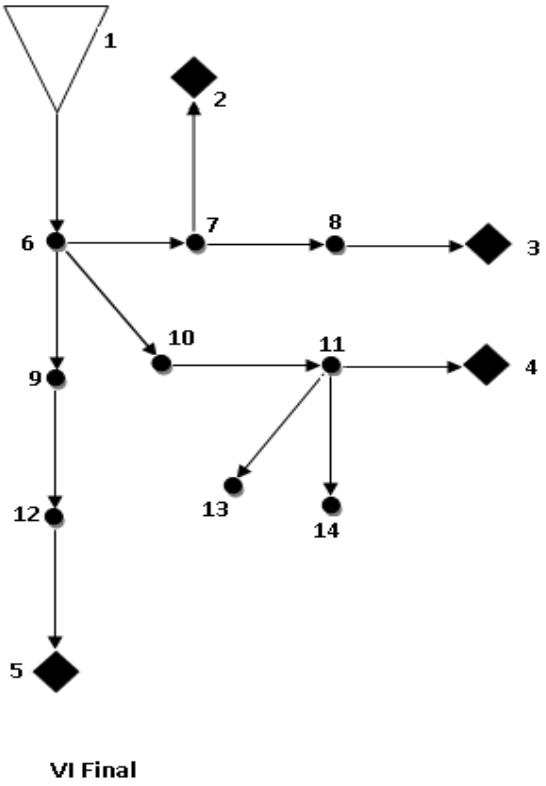


Figure 2 Tree stages

The final tree table emanating from node 1 in shown in table 5 below.

Node (j)	Total impedance to node j	Node preceding j
1	0	-
2	15	7
3	20	8
4	18	11
5	19	12
6	5	1
7	13	6
8	17	7
9	10	6
10	9	6
11	14	10
12	17	9
13	22	11
14	19	11

Table 5: Final tree table

7. Conclusion

The algorithm was tested successfully and was found in the case study that the minimum tree emanating from node 1 and terminating in all other nodes of the network. It does not contain any other paths. The sequence of the links does not represent minimum path between node 3 and node 14. However in order to find minimum tree emanating from node 3, the procedure needs to be repeated starting with appropriate initialization of the tree table. The results were found to be consistent with the expected results. The algorithm, which has been the focus of this entire paper, has thus been validated for the obstacles that were being considered critical, by the authors. By using this approach we were able to dramatically reduce the run time performance compared to conventional techniques while still maintaining an acceptable level of accuracy.

8. Acknowledgement

The authors would like to thank everyone.

9. References

[1] Dreyfus, S.E (1969) "An Appraisal of some shortest path algorithms". Journal Operations Research 17, 395-412

[2] Qiujin Wu, Joanna Hartley & David AL-Dabass, "Time Dependent Stochastic shortest paths algorithms for a scheduled transportation network".

[3] Yongtaek LIM & Hyunmyung KIM, "A shortest path algorithm for real road network based on path overlap", Journal of the Eastern Asia Society for Transportation studies, Vol 6, 1426-1438, 2005

[4] Qiujin Wu, Joanna Hartley & David AL-Dabass, "Using K shortest path algorithms to accommodate user preferences in the optimization of public transport travel".

[5] Brander A W, Sinclair M C 1995, "A comparative study of k shortest path algorithms", Proceedings of the 11th UK Performance Engineering workshop for Computer & Telecommunication Systems.

[6] Barbara Chapman, Gabriele Jost And Ruud Van Der Pas, "Using OpenMP - Portable Shared Memory Parallel Programming", The MIT Press, 2008 – Pages 15 to 107.

[7] Mohd Yamani Idna Idris, Emran Mohd Tamil, Noorzaily Mohamed Noor and Zaidi Razak, "Map Route Extraction and shortest path algorithm for public transport information system".

[8] B Horvath, "A New Public Transport Assignment Model", Acta Technica Jaurinensis Vol 1 No 1, 2008 Pages 93 to 108.

[9] Dickson K W Chiu, Oliver K F Lee, Ho-fung Leung, Eric W K Au and May C W Wong, " A Multi modal agent based mobile route advisory system for public transport network", Proceedings of the 38th Hawaii International Conference on System Sciences - 2005

[10] Hutchinson, B.G., "Route *Assignment Analysis*", Principle of Urban Transport Systems Planning, Mc Graw Hill Book Company, pp.118.

[11] Papacostas, C.S., "*Sequential Demand Forecasting Models*", Fundamentals of Transportation Engineering, Prentice-Hall Publishers, pp. 245 – 302.

[12] Aho, A., Hopcroft, J., and Ullman, J., "The design and analysis of computer algorithms". Addison-Wesley, 1974.

[13] Ahuja, R., Magnanti, T., and Orlin, J., "Network °ows: theory, algorithms and applications". Englewood Cli®s, NJ: Prentice Hall, 1993.

[14] U. Meyer and P. Sanders., "Stepping: A parallelizable shortest path algorithm". *Journal of Algorithms*, 2002. Accepted for publication.

[15] U. Meyer and J. F. Sibeyn. "Oblivious gossiping on tori". *Journal of Algorithms*, 42(1):1–19, 2002.

[16] U. Meyer and P. Sanders. "Stepping: A parallel single source shortest path algorithm". In *Proc. 6th Ann. European Symposium on Algorithms (ESA)*, volume 1461 of *LNCS*, pages 393–404. Springer, 1998.

Authors

Guruprasad Nagaraj is a Research Scholar with Dept. of Computer Science at MGR Deemed University, Chennai. Persuing his PhD in the field of Operations Research. He has over 18 yrs of experience in teaching undergraduate and post graduate students and currently working as Associate Professor in Dept. of Computer Science, PES Institute of Technology, Bangalore. His area of interest is Programming Languages, Data Structures & Algorithms, UNIX Operating system and Operations Research.

Y S Kumaraswamy is currently a Senior Professor and Head – Master of Computer Applications at Dayanand Sagar College of Engineering, Bangalore. He obtained his Post Doctoral Degree from Indian Institute of Science. Currently guiding PhD scholars in the field of Operations Research, Mathematics and Computer Science.