Shehzad, Farrukh; Shah, Muhammad Akbar Ali

Article

Evaluation of shortest paths in road network of Sindh-Pakistan

Pakistan Journal of Commerce and Social Sciences (PJCSS)

Provided in Cooperation with:
Johar Education Society, Pakistan (JESPK)

Suggested Citation: Shehzad, Farrukh; Shah, Muhammad Akbar Ali (2009) : Evaluation of shortest paths in road network of Sindh-Pakistan, Pakistan Journal of Commerce and Social Sciences (PJCSS), ISSN 2309-8619, Johar Education Society, Pakistan (JESPK), Lahore, Vol. 3, pp. 67-79

This Version is available at:
https://hdl.handle.net/10419/187996

Evaluation of Shortest Paths in Road Network

Farrukh Shehzad (Corresponding author)
National College of Business Administration and Economics, Lahore, Pakistan
E-mail: fshehzad.stat@gmail.com

Muhammad Akbar Ali Shah
Department of Statistics, The Islamia University of Bahawalpur, Pakistan
E-mail: akbar_alishah@yahoo.com

Abstract

Optimization is a key factor in almost all the topics of operations research / management science and economics. The road networks can be optimized within different constraints like time, distance, cost and traffic running on the roads.

This study is based on optimization of real road network by means of distances. Two main objectives are pursued in this research: 1) road distances among different routes are composed in detail; 2) two standard algorithms (Dijkstra and Floyd-Warshall algoritms) are applied to optimize/minimize these distances for both single-source and all-pairs shortest path problems.

Keywords: Shortest Path, Dijkstra Algorithm, Floyd-Warshall Algorithm, Road Networks

1. Introduction

It is, perhaps, by nature that human beings mostly emphasize optimization in real life phenomena. Finding the shorter times and the lower costs are not only looked upon by individuals but also by multinational companies. For an individual, it is often only a matter of convenience, but for a corporation it is of strategic importance when direct monetary cost is involved. The shortest path problem is one of the most important optimization problems in such fields as operations research / management science, computer sciences and artificial intelligence. One of the reasons is that essentially any combinatorial optimization problem can be formulated as a shortest path problem (Sniedovich, 2005). The development of algorithms for this shortest path finding problem, their computational testing and efficient implementation have remained important research topics within related disciplines (Dijkstra (1959), Dial et al. (1979), Glover et al. (1985), Ahuja et al. (1990), Goldberg and Radzik (1993)).

In this study, we have optimized one real road network in terms of distances using two numerous algorithms: 1) Dijkstra algorithm and 2) Foyd-Warshall algorithm. The review of graph theory related to the topic, shortest path problem, shortest path algorithms and the network related data (edge-wise distances) with computational procedure, results and conclusion are arranged in the later sections.

2. Graph Theory Context

A network is defined as a directed graph G = (N, A) consisting of a set N of nodes (or vertices) and a set A of arcs, as the number of nodes, n = $|N|$, and the number of arcs, m = $|A|$. The arcs of a network are links (or lines or branches) that connect certain pairs of nodes. If the flow of an arc is unidirectional or there is a one-way link between two nodes, the arc is said to be a directed arc. Whereas, if an object can move in either direction of an arc, it is said to be an undirected arc (or an edge). A network that has only directed arcs is known as directed network. In the same way, an undirected network is one that has all its arcs undirected.

In a network, each arc or edge is associated with some numerical value (real number), variously called its cost, weight, length (distance) or some other variable depending on the application. It is denoted as c_{ij} for each arc $(i, j) \in$ A. Weights of the arcs are very important because some algorithms impose further restrictions on weights. A path is a sequence of distinct arcs/edges connecting two specified nodes in a network. Each arc/edge must have exactly one node in common with its predecessor in the sequence and no node may be visited more than once (Rardin, 2003). A path may be either directed or undirected. If a path begins and ends at the same node it is called a cycle. In an undirected network there are many cycles. However, note that the definition of path must satisfy. A directed network may or may not be cyclic depending on whether the paths involved are directed or undirected. If a network has at least one cycle/directed cycle then it refers to as a cyclic network otherwise called acyclic. By the same token, a negative cyclic network is a network which has at least one cycle of negative total length (negative cycle).

The shortest path is a connected path between any two nodes with least weights (or costs). This problem can be stated as: given a network, find the shortest paths (or shortest distances or least costs) from a source node *s* to other node(s) or shortest path between every two nodes. The shortest paths from a source node to all others (one-to-all) represent a directed tree T rooted from source node *s*, with the characteristic that a unique path from *s* to any node i in the network is the shortest path to that node (Zhan and Noon, 1996). This directed tree is called a shortest path tree. For any network with n nodes, one can obtain n distinct shortest path trees. The length of the shortest path from *s* to any node i is denoted as $d(i)$.

3. Shortest Path Problem

Finding the shortest path is an important task in many network and transportation related analyses. This problem arises as a main decision question or as a step in some situation. There are many variations, depending on the type of network and costs involved, and source/destination pairs of nodes for which we need solution (Rardin, 2003).

It is therefore important to determine the specificity of the shortest path problem we are concerned. The most in depth classification of shortest path problems is agreed to the Deo and Pang's taxonomy (1984) and most recently by Sniedovich (2005).

Several variants of the shortest path problem are summarized as under:

- **Cyclic or Acyclic problems:** If there is at least one cycle in the network it is called a cyclic network otherwise acyclic.
- **Non-negative or Negative distance problems:** If the distances are non-negative, that is, $c_{ij} \geq 0$ for all *i* and *j*, or if there is at least one negative distance, that is, $c_{ij} < 0$.
- **Non-negative cyclic or Negative cyclic problems:** If the cyclic problems have non-negative length of all cycles or if the length of at least one cycle is negative (Sniedovich, 2005).
- **Sparse or Dense network problem:** A network where m, number of arcs, is closer to n^2 ('n' number of nodes) is a dense network, whereas a graph where m = (Alpha × n) where alpha is much smaller than n is a sparse network (that is, when the arc-to-node ratio is small).
- **Single-source or All-pairs shortest path problem:** Whether to find the shortest path from one source node to one destination node (one-to-one), shortest path from one source node to a subset of nodes (<n) (one-to-some) or one source node to all other nodes (one-to-all).

The presence of negative arc lengths allow negative cycles, that is, cycles with negative lengths. If such a negative cycle exists the shortest path algorithms whether of dynamic programming based or linear programming based have the solution unbounded. The networks in this study, for which we have to determine optimum distances and optimum paths are undirected, cyclic, sparse and have non-negative arc lengths.

4. Shortest Path Algorithms

A variety of methods and algorithms are available for the solution of shortest path problems depending on the nature of specific problem. There are of course, a number of ways to classify such algorithms like: i) dynamic programming inspired algorithms and linear programming inspired algorithms and ii) label setting algorithms and label correcting algorithms. The first classification scheme is very much OR/MS (operations research / management science) and methodologically oriented and the second classification scheme used both in computer science and operations research literature [Bertsekas (1991), Evans and Minieka (1992)] (Sniedovich, 2005).

Because of the nature of our problem, we just review the two dynamic programming algorithms; (i) Dijkstra algorithm and (ii) Floyd-Warshall algorithm. These dynamic programming algorithms of shortest path problem are inspired by the famous Bellman's (1957) principle of optimality and are the procedures designed to find shortest path by solving the dynamic programming functional equations. Detailed description of these algorithms, with modifications and experimentations using different data structures, is given in Cherkassky et al. (1993) and Zhan and Noon (1996) and references therein.

4.1 Dijkstra's Algorithm

In 1959 Edsger Wybe Dijkstra, a Dutch computer scientist, proposed two algorithms for the solution of two fundamental graph theory problems: i) the minimum weight spanning tree problem and ii) the shortest path problem.

Sniedovich (2005) comments that the original Dijkstra's algorithm constitutes an iterative procedure for the solution of dynamic programming functional equation associated with the shortest path problem given that the arc lengths are non-negative.

Dijkstra's algorithm for the single-source (i.e. one-to-one, one-to-some and one-to-all) shortest path problem is one of the most celebrated algorithms in computer science and a very popular algorithm in operations research. There have been many refinements and modifications as well as numerical experiments (evaluations) that have brought a significance improvement in the performance of the algorithm especially due to the use of some new data structures.

We take up the modified form of Dijkstra algorithm by Rardin (2003) and further make some modifications in the sense that Rardin (2003) gave a four step algorithm to calculate shortest distances and then a backtracking procedure to recover the shortest paths. But here in this study we modified these two tasks in ten step algorithm. In addition these ten steps can be applied for all subclasses of single-source shortest path problem.

Dijkstra's algorithm forms two different tables (say table A and table B on page 9-10) that summarize the calculations and updates of iterations. The final results of table A represent the shortest distances from source node *s* to all other nodes whereas table B helps in recovering the corresponding shortest paths.

The steps of this algorithm are given below:

Step 1: Select the source node'*s*' and initialize the optimal path lengths in first row of table A with *s* as

$$d[i] \leftarrow \begin{cases} 0 & \text{if } i = s \\ \infty & \text{otherwise} \end{cases}$$

Also start table B with $u[i]$ instead of $d[i]$.

Step 2: Select $p \leftarrow s$ as the first permanently labeled node and add it in column of permanently labeled nodes (first column) of table A and B.

Step 3: For every arc/edge (p, i) emanating from node p, update

$$d[i] \leftarrow \min \{ d[i], d[p] + c_{p,i} \} \qquad (4.3)$$

And mark node p permanent.

Step 4: If $d[i]$ changed in value, set $u[i] \leftarrow p$ in table B.

Step 5: Check whether the recently permanent node is

- the destination node (if the problem is one-to-one)
- last one of which the shortest path are required (if the problem is one-to-some)
- last node to be permanent or no temporary node remain(if the problem is one-to-all)

If so, go to step 7; otherwise go to step 6.

Step 6: Select p as the next permanently labeled node with least current value $d[i]$, that is, $d[p] = \min \{ d[i] : i \text{ temporary} \}$. And go to step 3.

Step 7: Mention node i (destination node) for which shortest path from source node s is required and prefix node i as the last node in the shortest path.

Step 8: Check whether the recently prefixed node is the required source node. If no, let this node be X and go to step 9; otherwise go to step 10.

Step 9: Take final $u[X]$ from table B and prefix it in the partially formed shortest path. Then go to step 8.

Step10: The required shortest path from source node s to node i is constructed and the corresponding shortest distances s to node i is the final $d[i]$ in table A. Note that the arcs/edges in this path form the shortest path tree by distinguishing to others.

If shortest path from source node to any other node (whose shortest distances is calculated) is required, go to step 7; otherwise stop.

4.2 Floyd-Warshall Algorithm

The Floyd-Warshall algorithm is a dynamic programming algorithm to solve the all-pairs (or all-to-all) shortest path problem on a directed network. The arcs of the network may have negative costs but must not have any negative-costs cycles. The modification for Floyd-Warshall algorithm is also made as in section 4.1. The purpose is to get the shortest distances and paths in a same sequence of steps. The four step Floyd-Warshall algorithm is also taken from Rardin (2003).

The Floyd-Warshall algorithm takes as input an adjacency matrix representation. This algorithm maintains two types of matrices, i) distance matrix D^t and ii) precedence matrix U^t, in each iteration and takes initial distance matrix D^0 and initial precedence matrix U^0 as input. Then proceeds for n iterations, where n is the number of nodes in the distance matrix. The n^{th} iteration gives the optimal/final distance matrix $D^{t=n}$ and the final precedence matrix $U^{t=n}$. The optimal distance matrix D^n represents the shortest distances between any two nodes in the network and the corresponding shortest paths can be traced out from the final precedence matrix U^n. The steps under Floyd-Warshall algorithm are summarized below:

Step 1: Form the initial distance matrix D^0 and the initial precedence matrix U^0.

For all arcs/edges (k,l) in the network, initialize

$$d^{(0)}[k,l] \leftarrow c_{k,l} \quad , \qquad u[k,l] \leftarrow k$$

For k, l pairs with no arc/edge (k,l), assign

$$d^{(0)}[k,l] \leftarrow \begin{cases} 0 & if\ k = l \\ +\infty & otherwise \end{cases}$$

Step 2: Set iteration counter t ← 1.

Step 3: Find values of distance matrix D^t for all k, $l \neq$ t using the relation below

$$d^{(t)}[k,l] \leftarrow \min \left\{ d^{(t-1)}[k,l], d^{(t-1)}[k,t] + d^{(t-1)}[t,l] \right\}$$

Step 4: Assign the values to precedence matrix U^t using the relation below

$$u[k,l] \leftarrow \begin{cases} u^t[t,l] & if \quad d^t[k,l] \neq d^t[k,l] \\ u[k,l] & otherwise \end{cases}$$

Step 5: Terminate if

i) $t = n$ 'the number of nodes in the network'

ii) $d^t[k,k] < 0$ for any node k

In first case $d^t[k,l]$ equal the required shortest distances and the latter one detects if there is any negative cycle through k.

Step 6: If t < n and all $d^t[k,k] \geq 0$, set $t \leftarrow t+1$ and go to step 3.

Step 7: To recover the shortest path between any two nodes (k,l), follow the guidelines given below:

7.1: Take node l as the last node in the shortest path.

7.2: Find the value $u[k,l]$ form the final precedence matrix U^n. Let it be X. Prefix node X in the partially formed shortest path.

7.3: Check whether X is equal to k. if so, go to step 7.4; otherwise set $l = X$ and go to step 7.2.

7.4: The required shortest path from node k to node l is constructed.

4.3 Some Evaluations of Shortest Path Algorithms

There have been a number of reported evaluations of shortest path algorithms in the literature e.g. Glover et al. (1985), Gallo and Pallottino (1988), Mondou et al. (1991), Cherkassky et al. (1993), Zhan and Noon (1996), Shad et al. (2003).

4.3.1 Shortest Path Algorithms: An evaluation using real road networks by Zhan and Noon (1996):

Zhan and Noon (1996) tested 15 of the 17 shortest path algorithms, which were tested by Cherkassky et al. (1993), using 21 real road networks.

They suggested that best performing implementations for solving the one-to-all shortest path problem are PAPE and TWO-Q for both large and small networks. Furthermore, if it is only necessary to compute a one-to-one shortest path or the shortest paths from a source node to a subset of the nodes (one-to-some), it may be worthwhile to consider one of the Dijkstra's implementations.

4.3.2 Evaluation of route finding methods in GIS application by Shad et al. (2003):

More recently, Shad et al. (2003) evaluated the processing time of a set of three shortest path algorithms using Visual Basic, Mapobject and Visual C++ programming language. These three algorithms are Dijkstra's

algorithm, Heuristic Methods and Genetic algorithms. They tested 4 shortest path algorithms using 6 real road networks. The 4 algorithms were implemented using data sets with different numbers of nodes and links. Finally, they suggested that none of the research regarding the evaluation of the performance of shortest path algorithms has a clear answer as to which algorithm or a set of algorithms runs fastest on real road networks.

5. Computation and Results

Data of road network of Sindh is composed in this study. This road network, which is characterized as real road network, consists of main towns and five level of roads i.e. motorways (R-1), national highways (R-2) metalled main (R-3), metalled other(R-4) and unmetalled (R-5) roads. The network contains 17 main towns of Sindh province.

As far as the sources of data are concerned, the distances of roads are taken from NHA, Surveyor General of Pakistan or approximate distances have manipulated directly from the road map where ever necessary. Table 5.1 describes the edge-wise detail of distances in road network of Sindh. Columns 4-8 show that the particular arc has the length of each type of road if included. The entries of total distance have a label, either I or II, with them. Label-I indicates that the distance is taken from NHA or road map, whereas label-II shows that approximate distances are manipulated from road map by a given scale.

The computational tables of the one network for both single-source (see table 5.2 and 5.3) and all-pairs problem (see tables 5.4 and 5.5) are given on next pages.

Figure . Roadmap of Sindh (Source: Highway Department)

Table 5.1 Edge-wise Approximate Distances in Road Network of Sindh

Edge No.	From	To	R-1	R-2	R-3	R-4	R-5	Total	Route
1	Karachi	Thatta		101				101-I	Karachi-Thatta
2	Karachi	Hyderabad		175				175-I	Karachi-Kotri-Hyderabad
3	Karachi	Dadu		340				340-I	Karachi-Kotri-Sehwan-Dadu
4	Thatta	Badin			93			93-II	Thatta-Sujawal-Badin
5	Thatta	Hyderabad		98				98-I	Thatta-Hyderabad
6	Badin	Mithi				58	52	110-II	Badin-Tando Bago-Jhudo-Naukot-Mithi
7	Badin	Hyderabad			100			100-I	Badin-Matli-Hyderabad
8	Badin	Mirpur Khas			52	86		138-II	Badin-Matli-Digri Mirwah-Mirpur Khas
9	Badin	Umarkot			52	123		175-II	Badin-Matli-Digri Mirwah-Umarkot
10	Mithi	Hyderabad			7	115	32	154-II	Mithi-Jhudo-Digri-Shaikh Bhirklo-Hyderabad
11	Mithi	Mirpur Khas				88	32	120-II	Mithi-Jhudo-Mirwah-Mirpur Khas
12	Mithi	Umarkot				20	95	115-II	Mithi-Jhudo-Nabisar-Umarkot
13	Hyderabad	Mirpur Khas			66			66-I	Hyderabad-Tando Allahyar-Mirpur Khas
14	Hyderabad	Sanghar		48		56		104-II	Hyderabad-Hala-Shahdadpur-Sanghar
15	Hyderabad	Nawabshah		89	24			113-I	Hyderabad-Hala-Sakrand-Nawabshah
16	Hyderabad	Naushahro Firoz		182				182-I	Hyderabad-Sakrand-Moro-Naushahro Firoz

17	Hyderabad	Dadu		180				180-I	Hyderabad-Kotri-Sehwan-Dadu
18	Mirpur Khas	Umarkot			72			72-II	Mirpur Khas-Pithoro-Umarkot
19	Mirpur Khas	Sanghar			63			63-I	Mirpur Khas-Sanghar
20	Sanghar	Nawabshah			61			61-I	Sanghar-Khadro-Nawabshah
21	Nawabshah	Naushahro Firoz				80		80-II	Nawabshah-Pad Idan-Naushahro Firoz
22	Nawabshah	Dadu		70	22	22		114-II	Nawabshah-Sakrand-Moro-Dadu
23	Nawabshah	Khairpur		31		112		143-II	Nawabshah-Pad Idan-Kot Diji-Khairpur
24	Naushahro Firoz	Dadu		26	22			48-II	Naushahro Firoz-Moro-Dadu
25	Naushahro Firoz	Khairpur		113				113-I	Naushahro Firoz-Ranjpur-Khairpur
26	Dadu	Larkana		110				110-II	Dadu-Mehar-Larkana
27	Larkana	Shikarpur		63				63-II	Larkana-Garhi Yasin-Shikarpur
28	Larkana	Jacobabad		20		80		100-II	Larkana-Ratodero-Jacobabad
29	Khairpur	Sukkur		23				23-I	Khairpur-Sukkur
30	Sukkur	Shikarpur		37				37-I	Sukkur-Shikarpur
31	Sukkur	Ghotki		58				58-I	Sukkur-Pano Aqil-Ghotki
32	Shikarpur	Jacobabad		43				43-I	Shikarpur-Jacobabad

Source: Highway Department

Table 5.2 (Table A) Single-source Shortest Path Solution of Sindh Road Network with Source Node 1(Karachi) by Dijkstra Algorithm

	p	*d*[1]	*d*[2]	*d*[3]	*d*[4]	*d*[5]	*d*[6]	*d*[7]	*d*[8]	*d*[9]	*d*[10]	*d*[11]	*d*[12]	*d*[13]	*d*[14]	*d*[15]	*d*[16]	*d*[17]
T	Initial	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
1	1	**Perm**	101			175						340						
2	2		**Perm**	194														
3	5				329	**Perm**	241		279	288	357							
4	3			**Perm**	304			369										
5	6						**Perm**	313										
6	8								**Perm**									
7	9									**Perm**				431				
8	4				**Perm**													
9	7							**Perm**										
10	11											**Perm**	450					
11	10										**Perm**							
12	13													**Perm**	454			
13	12												**Perm**			513	550	
14	14														**Perm**	491		512
15	15															**Perm**	534	
16	17																	**Perm**
17	16																**Perm**	
	Final	0	101	194	304	175	241	313	279	288	357	340	450	431	454	491	534	512

Table 5.3 (Table B) For Recovering the Shortest Paths by Dijkstra Algorithm

t	*p*	*u*[1]	*u*[2]	*u*[3]	*u*[4]	*u*[5]	*u*[6]	*u*[7]	*u*[8]	*u*[9]	*u*[10]	*u*[11]	*u*[12]	*u*[13]	*u*[14]	*u*[15]	*u*[16]	*u*[17]
1	1		1			1						1						
2	2			2														
3	5				5		5		5	5	5							
4	3				3			3										
5	6							6										
6	8																	
7	9													9				
8	4																	
9	7																	
10	11												11					
11	10																	
12	13														13			
13	12															12	12	
14	14															14		14
15	15																15	
16	17																	
17	16																	
	Final		1	2	3	1	5	6	5	5	5	1	11	9	13	14	15	14

Table 5.4 Optimal/Final Distance Matrix by Floyd-Warshall Algorithm [All-to-all Shortest Path Solution of Sindh Road Network]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	101	194	304	175	241	313	279	288	357	340	450	431	454	491	534	512
2	101	0	93	203	98	164	236	202	211	280	278	388	354	377	414	457	435
3	194	93	0	110	100	138	175	201	213	282	280	390	356	379	416	459	437
4	304	203	110	0	154	120	115	183	244	324	334	444	387	410	447	490	468
5	175	98	100	154	0	66	138	104	113	182	180	290	256	279	316	359	337
6	241	164	138	120	66	0	72	63	124	204	238	348	267	290	327	370	348
7	313	236	175	115	138	72	0	135	196	276	310	420	339	362	399	442	420
8	279	202	201	183	104	63	135	0	61	141	175	285	204	227	264	307	285
9	288	211	213	244	113	124	196	61	0	80	114	224	143	166	203	246	224
10	357	280	282	324	182	204	276	141	80	0	48	158	113	136	173	216	194
11	340	278	280	334	180	238	310	175	114	48	0	110	161	184	173	210	242
12	450	388	390	444	290	348	420	285	224	158	110	0	123	100	63	100	158
13	431	354	356	387	256	267	339	204	143	113	161	123	0	23	60	103	81
14	454	377	379	410	279	290	362	227	166	136	184	100	23	0	37	80	58
15	491	414	416	447	316	327	399	264	203	173	173	63	60	37	0	43	95
16	534	457	459	490	359	370	442	307	246	216	210	100	103	80	43	0	138
17	512	435	437	468	337	348	420	285	224	194	242	158	81	58	95	138	0

Table 5.5 Final Precedence Matrix by Floyd-Warshall Algorithm [All-to-all Shortest Path Solution of Sindh Road Network]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	-	1	2	3	1	5	6	5	5	5	1	11	9	13	14	15	14
2	2	-	2	3	2	5	6	5	5	5	5	11	9	13	14	15	14
3	2	3	-	3	3	3	3	6	5	5	5	11	9	13	14	15	14
4	2	3	4	-	4	4	4	6	8	9	5	11	9	13	14	15	14
5	5	5	5	5	-	5	6	5	5	5	5	11	9	13	14	15	14
6	5	5	6	6	6	-	6	6	8	9	9	11	9	13	14	15	14
7	5	5	7	7	6	7	-	6	8	9	9	11	9	13	14	15	14
8	5	5	6	6	8	8	6	-	8	9	9	11	9	13	14	15	14
9	5	5	5	6	9	8	6	9	-	9	9	11	9	13	14	15	14
10	5	5	5	6	10	8	6	9	10	-	10	11	10	13	14	15	14
11	11	5	5	5	11	8	6	9	11	11	-	11	10	13	12	12	14
12	11	5	5	5	11	8	6	9	11	11	12	-	14	15	12	12	14
13	5	5	5	6	9	8	6	9	13	13	10	15	-	13	14	15	14
14	5	5	5	6	9	8	6	9	13	13	10	15	14	-	14	15	14
15	5	5	5	6	9	8	6	9	13	13	12	15	14	15	-	15	14
16	5	5	5	6	9	8	6	9	13	13	12	16	14	15	16	-	14
17	5	5	5	6	9	8	6	9	13	13	10	15	14	17	14	15	-

Results

In single-source problem, one-to-one and one-to-some problems can be obtained by deducing the one-to-all shortest path problem when the required node(s) became permanently labeled. Moreover, due to the property of reducing the one-to-all problem upto one-to-one problem, the networks are analyzed for one-to-all problem under single-source shortest path problem class only.

Setting Karachi (node 1) as a source city, table 5.2 and 5.3 represent the optimum distances and their corresponding paths from Karachi to all other cities in road network of Sindh.

Interpretation: The required optimum/shortest distance from Karachi to any other city in the network is the final value of the column labeled by an integer in table 5.2 (table A). For example, if the shortest path from Karachi to Sukkur (node 14) is required, the final value of column *d*[14] i.e. 454 Km is the shortest distance. On the other hand the corresponding shortest route recovered by following the steps 7-10 of Dijkstra algorithm in section 4.1 is 1-5-9-13-14 i.e. Karachi-Hyderabad-Nawabshah-Khairpur-Sukkur. The same mechanism can be applied to get the shortest path to any other city from Karachi. The other subclasses of the shortest path problem, one-to-one and one-to-some problems, can also be managed through this one-to-all shortest path problem. For example, in calculating the shortest path from Karachi to Sukkur only, the algorithm will reduce upto 14^{th} iteration because Sukkur (node 14) became permanently labeled in 14^{th} iteration.

The solution of all-to-all shortest path problem in road network of Sindh by applying the Floyd-Warshall Algorithm is given in table 5.4 and 5.5.respectively. The (*i*,*j*)element of the final distance matrix is the shortest or optimal distance from node *i* to node *j*. For example the shortest distance from Hyderabad (node 5) to Shikarpur (node 15) is equal to the (5, 15) element of final distance matrix i.e. 316 Km. and its corresponding route recovered from final precedence matrix is 5-9-13-14-15 i.e. Hyderabad-Nawabshah-Khairpur-Sukkur-Shikarpur.

6. Conclusion

Finding the shortest path is often a central task in many network and transportation analysis problems. The main reasons are: many of the optimization problems can be formulated as networks and it is usually be the essential part of more complicated transportation analysis problems. Considerable research has been done to develop the faster algorithms for solving this problem and of course, there have been numerous algorithms like Bellman-Ford-Moore algorithm, Dijkstra algorithm etc. Moreover, some important evaluations have also made to test the efficiency of these algorithms.

Unfortunately these evaluations were usually based on randomly generated networks which cannot reflect the characteristics of real road networks that we are concerned. Zhan and Noon (1996) and Shad et al. (2003) have evaluated the shortest path algorithms using real road networks. However, these evaluations are quite worthy in their class.

This is basically an empirical study and we have pursued two main objectives:

1. Generated detailed data related to road distances.
2. Solved the shortest path problem through standard algorithms (Dijkstra and Floyd-Warshall)

The choice of Dijkstra algorithm is due to its characteristic of running just on non-negative arc length, solving all the subtypes of single-source shortest path problem and taking any of the nodes as source. Its modification also justify Bellman's principal of optimality that shortest path has the shortest sub-paths.

The second algorithm, Floyd-Warshall algorithm, is also most generally used to find the shortest distance from any node to any other node in networks. See tables 5.4 to 5.5 for the final results of our network using this algorithm.

Finally the results of our study might be a guideline not only for passengers who travel by roads but also for transporters and for the companies that spend a lot on road transportation. Government agencies such as Highway Department may use this methodology to prioritize its limited budget to construct certain routes in order to best utilize the scare national resources.

References

Ahuja, R. K., Mehlhorn, K., Orlin, J. B., and Tarjan, R. E. (1990). Faster Algorithms for the Shortest Path Problem. *Journal of Association of Computing Machinery*, 37, 213-223.

Bellman, R. (1957). *Dynamic Programming*, Princeton University Press, Princeton, NJ.

Bertsekas, D. (1991). *Linear Network Optimization*, The MIT Press, Englewood Cliffs, NJ.

Cherkassky, B. V., Goldberg, A. V., and Radzik, T. (1993). Shortest Path Algorithms: Theory and Experimental Evaluation. Technical Report 93-1480, Computer Science Department, Stanford University.

Deo, N and Pang, C. (1984). Shortest Path Algorithms: Taxonomy and Annotation, *Networks*, 14, 275-323.

Dial, R. B., Glover, F., Karney, D., and Klingman, D. (1979). A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding Shortest Path Trees, *Networks*, 9, 215-248.

Dijksta, E. W. (1959). A Note on Two Problems in Connection with Graphs, *Numeriche Mathematik*, 1, 269-271.

Evans, J. R. and Minieka, E. (1992). *Optimization Algorithms for Networks and Graphs*, Marcel Dekker, NY.

Gallo, G. and Pallotino, S. (1988). Shortest Path Algorithms, *Annals of Operations Research*, 13, 3-79.

Glover, F., Klingman, D., and Philips, N. (1985). A New Polynomial Bounded Shortest Paths Algorithm. *Operations Research*, 33, 65-73.

Goldberg, A. V., and Radzik, T. (1993). A Heuristic Improvement of the Bellman- Ford Algorithm, *Applied Mathematics Letter*, 6, 3-6.

Mondou, J. H., Crainic, T. G., and Nguyen, S. (1991). Shortest Path Algorithms: A Computational Study with the C Programming Language. *Computers & Operations Research*, 18, 767-786.

Rardin, R. L. (2003). *Optimization in Operations Research*, Pearson Education, Delhi, India.

Shad, R., Ebadi, H. and Ghods, M. (2003). Evaluation of Route Finding Methods in GIS Application, Presented in Map Asia Conference 2003.

Sniedovich, M. (2005). Dijkstra's Algorithms revisited: The OR/MS Connextion, Department of Mathematics and Statistics, The University of Melbourne, Parkville VIC 3052, Australia.

Zhan, F. B., and Noon, C. E. (1996). Shortest Path Algorithms: An Evaluation Using Real Road Networks. *Transportation Science*, 32, 65-73.