International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 9, September - 2013

A Study on Different Algorithms for Shortest Route Problem

Dr. Roopa, K.M., Apoorva, H.R1., Srinivasu,V.K.2 and Viswanatah, M.C 3

Abstract

Shortest path problems are among the most studied network flow optimization problems with interesting application across a range of fields. In this paper, three shortest path algorithms are discussed viz. Dijkstra's Algorithm (one to all pairs of nodes), Floyd Warshall's Algorithm (all to all pairs of nodes) and Linear Programming Problems (LPP). These algorithms are also solved using Matlab software, which gives quick results for larger nodes. This paper also deals with the methodology to find shortest distance using the dual of Linear Programming Problems. In addition, Complementary Slackness Theorem is discussed to solve the primal problem from the solution of dual problem and determine the shortest distance as well as shortest routes.

Key Words: Shortest Path, Dijkstra's Algorithm, Floyd Warshall's Algorithm, Linear Programming Problem, Algebraic Method, Matlab

1.0 Introduction:

Finding the shortest path is an important task in network and transportation related analysis. Shortest distance problems are inevitable in road network applications, such as city emergency handling and driving system, where optimal routing has to be found. Therefore, network optimization has always been the heart of operational research. Also, as traffic conditions of a city change from time to time, there could be a huge amount of request occurring at any moment, for which an optimal path solution has to be found quickly. Hence, efficiency of an algorithm is very important to determine the shortest routes are between nodes in a network.

There are many algorithms that can be used to determine the shortest route between two nodes in a network. In this paper, two standard algorithms Dijkstra's algorithm [1], [4] and Floyd Warshall's algorithm are discussed and also solved using Matlab software**. T**he linear programming formulation of shortest route problem solved using (0-1) binary integer programming technique is also discussed. The dual of formulated linear programming and shortest route problem [2] solved by algebraic method is demonstrated for small number of nodes, as it is difficult to solve for large number of nodes. In such cases, Matlab software can be the best choice. Further, the shortest distance and shortest route determined using Complementary Slackness Theorem [5].

2.0 Dijkstra's Algorithm:

Let G = (N, A), where, N = (1,2,3…..n) is a node, be a weighted and directed network in which the weight of the every directed arc (edge) is non negative. Dijkstra's algorithm is used to find the path with minimum weight from a chosen vertex, say vertex 1, to any other vertex 's' of G. This algorithm is iterative in nature and each of these iterations consists of two steps to calculate the shortest distance. Further, the algorithm provides means to trace the shortest path accordingly.

2.1 Outline of Dijkstra's Algorithm:

Dijkstra's algorithm considers two sets: i) set P, which at any specific point consists of all the nodes that were encountered by the algorithm and ii) set S, a precedence set, which at any specific point consists of the precedent node for each node in the network. Apart from these sets, the algorithm utilizes the following distance information.

$q_{ij,}$ for i, j=1, 2, 3, 4…n and i≠j, denote the weight of the directed edge (arc) from vertex i to vertex j. If there is no arc from i to j, then $q_{ij,}$ is set to be infinity.

t_j, for j=1, 2, 3, 4…n and j≠s where s is the start index. Also,

$$t_j = q_{1j} \quad \text{for } j=2,3,4\ldots n \qquad (1)$$

In each iteration, the sets P and S as well as the set of all t_j, for j=1, 2, 3, 4…n and j≠s, that are output from the previous iteration are taken as inputs.

Initially P = {s}. S is a set of size n populated with i) 0 if t_j = infinity, ii) s if t_j = finite value

The steps involved in each iteration for finding the shortest distance are summarized below:

Step 1: Identify minimum among the computed t_j values. Let t_k be the minimum.

Add k to the set P.

Step 2: Now P = {1, k}. For each of the nodes not in P and with finite q_{kj}, for j=1, 2, 3, 4…n and $j \notin P$, recalculate t_j using the below expression:

$$t_j = \min\{ t_{j,}\ t_k + q_{kj} \} \qquad (2)$$

Only if $(t_k + q_{kj}) < t_j$, then update the jth entry in S to k.

Continue the iterations until the end node, e, is added to the set P.

Similarly, the steps to trace the shortest path between nodes s and e, using Dijkstra's algorithm are given below:

Step 1: Take node e as the last node in the shortest path

Step 2: Find the eth entry in the set S, let this be x. Add this prefix node x to the partially constructed shortest path.

Step 3: Check whether x is equal to s. If so, go to Step 4; else go to Step 3.

Step 4: The required shortest path from node s to node e is thus constructed.

Example: Determine the shortest distance and shortest path from vertex 1 to vertex 7 in the weighted, directed network is depicted in a given Fig. 2.1.

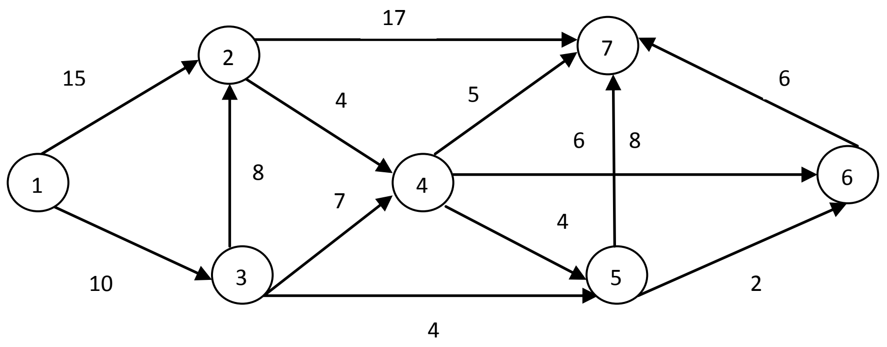

Figure 2.1: The shortest route network

First Iteration: P= {1}, S={0,1,1,0,0,0,0}, and

$t_2 = 15, t_3 = 10, t_4 = \infty, t_5 = \infty, t_6 = \infty, t_7 = \infty$

Step 1: Note that t_j is minimum for j=3, i.e., t_3 =10 .Therefore add 3 to the set P

Step 2: Now, P={1,3} and t_3 =10

$t_2 = \min\{t_2, t_2 + q_{32}\}$ => $t_2 = \min\{15, 10+ 8\} =15$

$t_4 = \min\{t_4, t_3 + q_{34}\}$ => $t_4 = \min\{\infty, 10 + 7\} =17$ => S [4] = 3

$t_5 = \min\{t_5, t_3 + q_{35}\}$ => $t_5 = \min\{\infty, 10+4\}= 14.$ => S [5] = 3

Second Iteration: P= {1, 3}, S={0,1,1,3,3,0,0} , and $t_2 = 15, t_3 = 10, t_4 = 17, t_5 = 14, t_6=\infty, t_7 =\infty$.

Step 1: For the set of recalculated t_j values, minimum occurs at j=5, i.e.t_5 =14. Therefore, add node 5 to set P.

Step 2: Now, P = {1, 3, 5} and t_5 =14

$t_6 = \min\{\infty, 14 + 2\} =16$ => S[6] = 5

$t_7 =\min\{\infty, 14+8\} = 22$ => S[7] = 5

Third Iteration: P = {1, 3, 5}, S={0,1,1,3,3,5,5} ,and $t_2 = 15, t_3 = 10, t_4 = 17, t_5 = 14, t_6 = 16, t_7 = 22$

Step 1: The minimum t_j value occurs for j=2 ,i.e. t_2 =15. Add node 2 to the set P.

Step 2: P= {1, 3, 5, 2} and t_2 =15

No change in the set of t_j values.

Fourth Iteration: P = {1, 3, 5, 2}, S = {0, 1, 1, 3, 3, 5, 5}, and $t_2 = 15, t_3 = 10, t_4 = 17, t_5 = 14, t_6 = 16, t_7 = 22$

Step 1: The minimum t_j value occurs for j=6 ,i.e. t_6 =16. Add node 6 to the set P.

Step 2: P = {1, 3, 5, 2, 6} and t_6 =16

t_7 = min {22, 16+6} =22 => since $t_7 = (t_6 + q_{67})$, S [7] remains unchanged.

Fifth Iteration: P = {1, 3, 5, 2, 6}, S= {0, 1, 1, 3, 3, 5, 5}, and $t_2 = 15$, $t_3 = 10$, $t_4 = 17$, $t_5 = 14$, $t_6 = 16$, $t_7 = 22$

Step 1: The minimum t_j value occurs for j=7, i.e., t_7 =22. Add node 6 to the set P. Add node 7 to the set P. As the end node, 7, is added to the set P, the process stops.

Thus, the shortest distance between node1 and node 7 = t_7 =22

The shortest route is obtained as follows:

S={0,1,1,3,3,5,5}and (s ,e)=(1,7).

In order to reach node 7 from node 1, take prefix node as node 5, S [7] = 5. Similarly, to reach from node 1 to node 5 take prefix node as node 3, S[5] = 3. Continuing this process, to reach from node 1 to node 3, take prefix node as node 1, S[3] = 1. As the prefix node, in this step, is same as the start node, this process stops. Thus, the shortest route is 1→3→5→7 which is of distance 22 units.

Other alternate routes from node 1 to node 7 are P={1,3,4,7} , P={1,3,5,6,7} which also gives shortest distance of 22 units.

The Dijkstra's algorithm to find the shortest path can also be solved using Matlab software and the corresponding output is given below:

dist =22 ; path =1 3 5 7; pred = 0 1 1 3 3 5 5

Thus, the shortest distance is 22 and shortest path is 1→3→5→7.

3.0 Floyd Warshall's Algorithm:

Floyd Warshall's Algorithm is a graph analysis algorithm to find the shortest route between any two nodes in a network with positive or negative edge weights with no negative cycle. This algorithm uses the dynamic programming technique to solve the shortest path problem between all pairs of nodes (all to all) in a directed network. It represents the network as a square matrix with n-rows and n- columns and at the end of the algorithm each (i,j) of the matrix gives the shortest distance from node i to node j. If there is a direct link between node i to node j, then the value at (i,j) is finite, otherwise it is infinite, i.e, d (i,j)= ∞.

Floyd Warshall's Algorithm is based on transitivity property. Given three nodes i, j and k are shown in Fig. 2.2.

From the transitivity property,

d (i ,j)+d(j,k)=d(i,k).

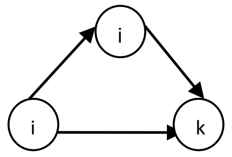

Figure 2.2: Transitivity Property

The algorithm exploits the above property to find the shortest distance between nodes i and k as follows:

If, d (i ,j)+d(j,k)<d(i,k). Then, it is optimal to replace the direct route from $i \rightarrow k$, where the indirect route is $i \rightarrow j \rightarrow k$.

3.2 Outline of Floyd Warshall's Algorithm:

The Floyd Warshall's algorithm uses two adjacency matrices i.e, i) distance matrix D_k and ii) precedence matrix S_k, where k=0, 1, 2…..n. In first iteration, the algorithm takes initial distance matrix D_0 and initial precedence matrix S_0 as input. Then on, in each iteration, the distance matrix and precedence matrix that are output from the previous iteration are taken as input. After n iterations, where n is the number of nodes in the distance matrix and the nth iteration gives the optimal/final distance matrix $D_{k=n}$ as well as the final precedence matrix $S_{k=n}$. 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 precedence matrix S_n. The steps for shortest distance under Floyd Warshall algorithm are summarized below:

Step1: Set iteration k=1.

Step2: Consider first column and first row of the initial representation of distance matrix D_0 as pivot column and pivot row and apply transitive operation.

Step3: If any entry of the pivot column or pivot row is infinity then row corresponding to this element need not be considered.

Step 4: There are two cases:

Case (a): If the condition $d(i,k)+d(k,j) < d(i,j)$ $(i \neq k, j \neq k, i \neq j)$, make the following changes

(i) Create D_k by replacing d (i, j) in D_{k-1} with d (i, k) + d (k, j)

(ii) Create S_k by replacing S_{ij} in S_{k-1}, set k=k+1 and repeat step k up to n steps .

Case (b): if $d(i,k)+d(k,j)=d(i,j)$ $(i \neq k, j \neq k, i \neq j)$; do not to make any changes, this implies that $i \rightarrow k \rightarrow j$ is an alternate route for $i \rightarrow j$.

Similarly, the steps to trace the shortest path between two nodes, say i and j using Floyd-Warshall's algorithm are given below:

Step 1: Take node *j* as the last node in the shortest path.

Step 2: Find the value *S* [i, j] from the precedence matrix $\mathbf{S_n}$, let it be x. Add this Prefix node x t partially constructed shortest path.

Step 3: Check whether x is equal to i. if so, go to step (4); else, set j = x and go to the step 3

Step 4: The required shortest path from node i to node j is constructed.

Example: Determine the shortest distance and shortest paths between all pairs of nodes in a transportation network as shown in Figure 2.1.

Iteration (0): Consider the initial representation of the matrix D_0 and S_0, as shown in fig 2.1.

d (i, j) = ∞, implies no traffic is allowed from node i to j.

D_0	1	2	3	4	5	6	7
1	-	15	10	∞	∞	∞	∞
2	∞	-	∞	4	∞	∞	17
3	∞	8	-	7	4	∞	∞
4	∞	∞	∞	-	4	6	5
5	∞	∞	∞	∞	-	2	8
6	∞	∞	∞	∞	∞	-	6
7	∞	∞	∞	∞	∞	∞	-

S_0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7
2	1	2	3	4	5	6	7
3	1	2	3	4	5	6	7
4	1	2	3	4	5	6	7
5	1	2	3	4	5	6	7
6	1	2	3	4	5	6	7
7	1	2	3	4	5	6	7

Iteration (1): Set k=1. Consider the first column and first row of D_0 as pivot column and pivot row respectively. As all the entries of the pivot column in D_0 are infinity, D_1 and S_1 are same as D_0 and S_0 respectively.

D_1	1	2	3	4	5	6	7
1	-	15	10	∞	∞	∞	∞
2	∞	-	∞	4	∞	∞	17
3	∞	8	-	7	4	∞	∞
4	∞	∞	∞	-	4	6	5
5	∞	∞	∞	∞	-	2	8
6	∞	∞	∞	∞	∞	-	6
7	∞	∞	∞	∞	∞	∞	-

S_1	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7
2	1	2	3	4	5	6	7
3	1	2	3	4	5	6	7
4	1	2	3	4	5	6	7
5	1	2	3	4	5	6	7
6	1	2	3	4	5	6	7
7	1	2	3	4	5	6	7

Iteration (2): Set k=2. Consider second column and second row of D_1 as pivot column and pivot row respectively. Except d (1, 2) and d (3, 2), all the entries in the pivot column are infinity. Also except d (2, 4) and d (2, 7), all the entries in the pivot row are infinity. Now apply transitivity property to obtain the following results:

(i) d(1,2)+d(2,4)= d(1,4)

But, d(1,2)+d(2,4)=15 + 4 and d(1,4) = ∞

This implies 19<∞, then d (1, 4) =19

d(1,2)+d(2,7)=d(1,7)

But, d(1,2)+ d(2,7)= 15 + 17 and d(1,7)=∞

this implies 32<∞, t hen d (1, 7) = 32

Similarly, 12>7, then d (3, 4) = 7 and 25<∞ , then d (3, 7) = 25.

Observe that there is no change in the value of d (3, 4) = 7. So, it cannot be improved.

(ii) The precedence matrix S1 can be changed as

S (1, 4) = 2, S (1, 7) = 2 & S (3, 7) = 2. These are the changes shown in the matrix D_2 and S_2

D_2	1	2	3	4	5	6	7
1	-	15	10	19	∞	∞	32
2	∞	-	∞	4	∞	∞	17
3	∞	8	-	7	4	∞	25
4	∞	∞	∞	-	4	6	5
5	∞	∞	∞	∞	-	2	8
6	∞	∞	∞	∞	∞	-	6
7	∞	∞	∞	∞	∞	∞	-

S_2	1	2	3	4	5	6	7
1	1	2	3	2	5	6	2
2	1	2	3	4	5	6	7
3	1	2	3	4	5	6	2
4	1	2	3	4	5	6	7
5	1	2	3	4	5	6	7
6	1	2	3	4	5	6	7
7	1	2	3	4	5	6	7

<u>Iteration (3)</u>: Set k=3. Consider third column and third row of D_3 as pivot column and pivot row respectively. Except d (1, 3), all the entries in the pivot column are infinity and also except d (3, 5) and d (3, 7), all the entries in the pivot row are infinity. Further, apply transitivity property to obtain the following results:

(i) Since, d(1,4)=17 , d(1,5)=14 and d(1,7)=32 . So, d (1,7) = 32 cannot be improved .

(ii) Set precedence matrix S_2 as S (1, 4) = 3, S (1, 5) = 3. The changes are as shown in the matrix D_3 and S_3

D_3	1	2	3	4	5	6	7
1	-	15	10	17	14	∞	32
2	∞	-	∞	4	∞	∞	17
3	∞	8	-	7	4	∞	25
4	∞	∞	∞	-	4	6	5
5	∞	∞	∞	∞	-	2	8
6	∞	∞	∞	∞	∞	-	6
7	∞	∞	∞	∞	∞	∞	-

S_3	1	2	3	4	5	6	7
1	1	2	3	3	3	6	2
2	1	2	3	4	5	6	7
3	1	2	3	4	5	6	2
4	1	2	3	4	5	6	7
5	1	2	3	4	5	6	7
6	1	2	3	4	5	6	7
7	1	2	3	4	5	6	7

Continuing in this way, the final matrix in the last iteration where none of the entries in the d (i,j) can be improved by transitivity property, because all the elements in the last row are infinity.

D_7	1	2	3	4	5	6	7
1	-	15	10	17	14	16	22
2	∞	-	∞	4	8	10	9
3	∞	8	-	7	4	6	12
4	∞	∞	∞	-	4	6	5
5	∞	∞	∞	∞	-	2	8
6	∞	∞	∞	∞	∞	-	6
7	∞	∞	∞	∞	∞	∞	-

S_7	1	2	3	4	5	6	7
1	1	2	3	3	3	5	4
2	1	2	3	4	4	4	4
3	1	2	3	4	5	5	4
4	1	2	3	4	5	6	7
5	1	2	3	4	5	6	7
6	1	2	3	4	5	6	7
7	1	2	3	4	5	6	7

Finally, the shortest distance between any two nodes is determined from the matrix D7.

Next to find the shortest path between nodes, say 1 and 7 i.e (i ,j)=(1,7). In order to reach node 7 from node 1, take prefix node as node 4, S_7 [1, 7] = 4. Similarly to reach from node 1 to node 4 take prefix node as node 3, S_7 [1, 4] = 3. Continuing this process to reach from node 1 to node 3, take prefix node as node 1, S_7 [1, 3] = 1. As the prefix node, in this step, is same as the start node, this process stops. Thus, the shortest route is 1$\rightarrow$3$\rightarrow$4$\rightarrow$7, which is of distance 22 units. Similarly, other possible shortest routes can be calculated from the matrix D_7 and S_7.

The Floyd Warshall's algorithm to find the shortest path (all too all pairs of nodes) can also be solved using Matlab software and the corresponding output is given below:

```
A=inf(7,7);   A(1,2)=15; A(1,3)=10;   A(2,4)=4;    A(2,7)=17;     A(3,2)=8; A(3,4)=7;    A(3,5)=4;

A(4,5)=4; A(4,6)=6; A(4,7)=5;    A(5,6)=2; A(5,7)=8;    A(6,7)=6;

>> [S,P,result] = FloydSPR(A,1,7)

S =                                   |    P =

 Inf   15   10   17   14   16   22    |      -1   -1   -1    3    3    3    3

 Inf  Inf  Inf    4    8   10    9    |      -1   -1   -1   -1    4    4    4

  Inf    8  Inf    7    4    6   12   |      -1   -1   -1   -1   -1    5    4

  Inf  Inf  Inf  Inf    4    6    5   |      -1   -1   -1   -1   -1   -1   -1

  Inf  Inf  Inf  Inf  Inf    2    8   |      -1   -1   -1   -1   -1   -1   -1

  Inf  Inf  Inf  Inf  Inf  Inf    6   |      -1   -1   -1   -1   -1   -1   -1

  Inf  Inf  Inf  Inf  Inf  Inf  Inf   |      -1   -1   -1   -1   -1   -1   -1

Result = 22.
```

4.0 Linear Programming Formulation of the Shortest Path Problem:

Linear Programming is a simple programming formulation problem. Most of the network problems can be formulated as Linear Programming Problems and can be solved using simplex method algorithm. In this section, two Linear Programming formulations for the shortest-route problem are discussed. These formulations are generally used to find the shortest route between any two nodes in the network.

Formulation 1: This formulation assumes that an external unit of flow enters the network at node s and leaves at node t, where s and t are the two nodes between which the shortest route is to be determined.

Define, x_{ij} = Amount of flow in arc (i , j), for all feasible i and j,

c_{ij} = Length of arc (i , j), for all feasible i and j.

Because only one unit of flow can be in any arc at any given time, the variable x_{ij} can assumes binary values (0 or 1) only. Thus, the objective function of the linear program becomes:

Minimize $Z = \sum_{\text{all defined arcs}} c_{i,j}\ x_{i,j}$,

The below constraint represents the conservation of flow at each node and for any node j;

Total input flow = Total output flow.

Formulation 2: This formulation is the dual problem of Linear Programming discussed in Formulation1. In the dual problem, the number of variables is equal to the number of nodes in the network. Also, it is equal to the number of constraints in formulation1. Further, all the dual variables must be unrestricted as all the constraints in Formulation1 are equations.

Let y_j be the dual constraint associated with node j. Given that s and t are the source and destination nodes of the network, then the dual problem is defined as follows:

Maximize $Z = y_t - y_s$

Subject to $y_j - y_i \leq c_{ij}$ for all feasible i and j. However, all y_i and y_j are unrestricted in sign.

Example: Consider the problem of determining shortest route in the network as shown in above Fig. 2.1. Here, s = 1 and t = 7. The below given Fig. 2.3 shows how a unit of flow enters at node1 and leaves at node 7.

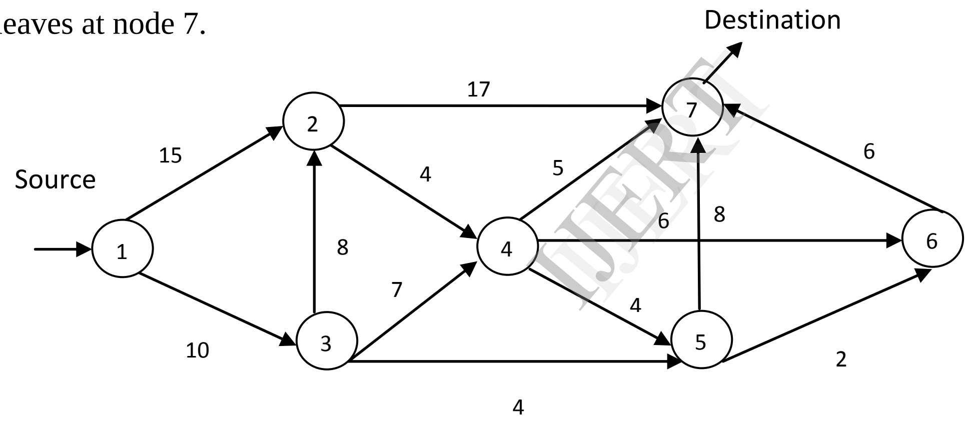

Figure 2.3: Network with source and destination

By setting $X_{i,j} = \begin{cases} -1 \text{ for node i} \\ 1 \text{ for node j} \end{cases}$ and then corresponding values of Linear program is listed below:

Min Z	X_{12}	X_{13}	X_{24}	X_{27}	X_{32}	X_{34}	X_{35}	X_{45}	X_{46}	X_{47}	X_{56}	X_{57}	X_{67}	
	15	10	4	17	8	7	4	4	6	5	2	8	6	
Node1	-1	-1												= -1
Node2	1		-1	-1	1									= 0
Node3		1			-1	-1	-1							= 0
Node4			1			1		-1	-1	-1				= 0
Node5							1	1			-1	-1		= 0
Node6									1		1		-1	= 0
Node7				1						1		1	1	= 1

From the above table, we obtain the following objective function and constraints of the linear programming problem as given below:

Min Z = $15x_{12}+10x_{13}+4x_{24}+17x_{27}+8x_{32}+7x_{34}+4x_{35}+4x_{45}+6x_{46}+5x_{47}+2x_{56}+8x_{57}+6x_{67}$

$$\text{Subject to}\quad \left.\begin{aligned} &-x_{12}-x_{13} = -1\\ &x_{12}-x_{24}-x_{27}+x_{32} = 0\\ &x_{13}-x_{32}-x_{34}-x_{35}=0\\ &x_{24}+x_{34}-x_{45}-x_{46}-x_{47} = 0\\ &x_{35}+x_{45}-x_{56}-x_{57} = 0\\ &x_{46}+x_{56}-x_{67} = 0,\ x_{27}+x_{47}+x_{57}+x_{67} = 1 \end{aligned}\right\} \qquad (3)$$

$x_{ij} \in \{0,1\}$ for i ,j=1,2,3,4,5,6,7

The above constraints represent the flow conservation at each node. Therefore, the problem is a binary integer programming problem. After solving this problem using Matlab software, the optimal solution Z=22 at $x_{13}=1$, $x_{34}=1$, $x_{47}=1$ obtained. This solution gives the shortest route from node 1 to node 7 as $1\rightarrow3\rightarrow4\rightarrow7$ and the associated distance is z = 22 units.

From the concept of dual of the Linear Programming Problem, the following objective function and constraints are obtained:

Maximize Z $=y_7-y_1$

Subject to y_i-y_j, for all feasible i and j

$y_2-y_1\le15$ (Route 1 to 2), $y_3-y_1\le10$(Route 1 to 3), $y_2-y_3\le8$ (Route 3 to2), $y_4-y_3\le7$ (Route 3 to 4), $y_5-y_3\le4$ (Route 3 to 5), $y_7-y_2\le17$(Route 2 to 7), $y_4-y_2\le4$ (Route 2 to4), $y_7-y_4\le5$ (Route 4 to 7), $y_6-y_4\le6$ (Route 4 to 6), $y_5-y_4\le4$(Route 4 to 5), $y_7-y_5\le8$ (Route 5 to7), $y_6-y_5\le2$ (Route 5 to 6), $y_7-y_6\le6$ (Route 6 to 7). Where, $y_1, y_2, \ldots, y_7$ are unrestricted in sign.

4.1 Algebraic Method for solving the dual Linear Programming Problem:

The dual linear programming problem can also be solved using algebraic method for only small number of variables. However, solving the above dual Linear Programming Problem through algebraic method, by introducing slack variables which gives better result compared to any other software packages. The first and final tableaus of algebraic method are given below:

First tableau:

NZV	y_1	y_2	y_3	y_4	y_5	y_6	y_7	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	Qty
S_1	-1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	15
S_2	-1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	10
S_3	0	1	-1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	8
S_4	0	0	-1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	7
S_5	0	0	-1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	4
S_6	0	-1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	17
S_7	0	-1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	4
S_8	0	0	0	-1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	5
S_9	0	0	0	-1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	6
S_{10}	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	4
S_{11}	0	0	0	0	-1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	8
S_{12}	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
S_{13}	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	6
Δ	1	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NZV: Non Zero Variables **Δ: Objective function with reversed sign**

Final tableau:

NZV	y_1	y_2	y_3	y_4	y_5	y_6	y_7	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	Qty
S_1	0	0	0	0	0	0	0	1	-1	0	0	-1	0	1	1	0	0	-1	0	0	2
y_3	-1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	10
S_3	0	0	0	0	0	0	0	0	0	1	0	-1	0	1	1	0	0	-1	0	0	5
S_4	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	1	0	0	-1	0	0	0
y_2	-1	1	0	0	0	0	0	0	1	0	0	1	0	-1	-1	0	0	1	0	0	13
S_6	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	-1	0	0	0	0	0	8
y_5	-1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	14
y_7	-1	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	1	0	0	22
S_9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	1	7
S_{10}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	1	1	0	0	7
y_6	-1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	-1	16
S_{12}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	1	0
y_4	-1	0	0	1	0	0	0	0	0	1	0	1	0	0	-1	0	0	1	0	0	17
Δ	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	22

From the above tableau, it is obtained that,y_1 =0, y_2 =13, y_3 =10, y_4 =17, y_5 =14, y_6 =16, y_7 =22. The above dual problem can also be solved using Matlab software. The solutions obtained from Matlab software are given below:

$y_1 = –10.4979$, $y_2 = 3.6415$, $y_3 = –0.4979$, $y_4 = 6.5021$, $y_5 = 3.5021$, $y_6 = 5.5021$, $y_7 = 11.5021$

The value of Z = 22 gives the shortest distance from node 1 to node 7. By considering, the solutions that satisfy the above constraints the following routes: 1-3, 3-4, 3-5, 4-7, 5-7, 5-6 and 6 -7 are obtained. From these sequence of routes 1-3, 3-4, 4-7, the shortest route $1\rightarrow 3\rightarrow 4\rightarrow 7$, which is of distance 22 units from node 1 to node 7 is traced. Similarly, other alternate shortest routes that can be obtained are: $1\rightarrow 3\rightarrow 5\rightarrow 7$ and $1\rightarrow 3\rightarrow 5\rightarrow 6\rightarrow 7$ respectively.

The shortest route can also be determined using Complementary Slackness Theorem [3] & [6]. As the sequence of routes 1-2, 3-2, 2-7, 2-4, 4-6 and 4-5, do not satisfy the constraints in the dual problem, from the Complementary Slackness Theorem it follows that $x_{12}=x_{32} = x_{24} =x_{27}=x_{45} =x_{46} = 0$

Substituting these variable values in the primal problem, the following systems of equations are obtained:

$$\left.\begin{aligned} &x_{13} = 1;\ \ x_{13}- x_{34}-x_{35} =0;\ \ x_{34}-x_{47}=0;\ \ x_{35}-x_{56}-x_{57} =0 \\ &x_{56}-x_{67} =0;\ \ x_{47}+x_{57}+x_{67} =1 \end{aligned}\right\} \qquad (4)$$

By solving above system of linear equations using Gauss Elimination Method, the system in echelon form becomes:

$$x_{34}+ x_{35} = 1;\ \ x_{35}+ x_{47} =1;\ \ x_{47}+x_{56} + x_{57} =1;\ \ x_{47}+x_{57} +x_{67} =1 \qquad (5)$$

In the above system of equations, there are 4 equations (r=4) with 6 unknowns (n=6) and two free variables (x_{35}, x_{56}). Hence, the possible choices are: (0,0),(0,1),(1,0),(1,1). Each of these possible choices may or may not be the solution points because the dependent variables have the restriction, $x_{ij} = 0$ or 1.

For the first choice (0, 0), i.e. $x_{35} =0$, $x_{56} =0$, by substituting these in equation (5), we get:

$x_{34} =1$, $x_{47} =1$, $x_{57} =0$, $x_{67} =0$, $x_{13} =1$. Then the shortest route is 1-3, 3-4 and 4-7 i.e. $1\rightarrow 3\rightarrow 4\rightarrow 7$.

Similarly, the third choice (1,0) and fourth choice (1,1) give the shortest routes which are $1\rightarrow 3\rightarrow 5\rightarrow 7$ and $1\rightarrow 3\rightarrow 5\rightarrow 6\rightarrow 7$ respectively. However, the second choice (0,1) does not give the shortest route because it does not satisfy the equation (5).

5.0 Conclusion:

From the below Fig. 2.4, it is evident that Dijkstra's algorithm takes a relatively lesser time than Floyds and Binary integer programming in finding shortest route. However, Dijkstra's algorithm is the better option for identifying the shortest path in larger networks such as railway, water, power distribution and gas pipeline networks.

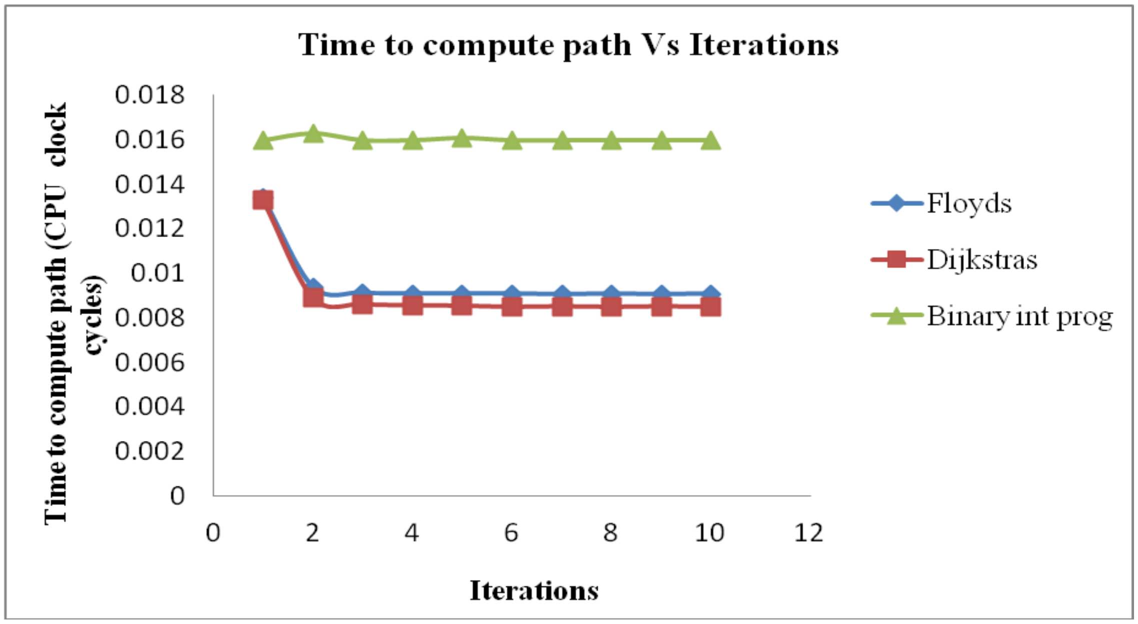

Fig.2.4: Comparison of computational time of various algorithms

Acknowledgement: The support from Dr.A.M. Ramesh, Senior Scientific Officer from Karnataka Science and Technology Academy is gratefully acknowledged.

References:

1. Dijkstra, E. W., 1959. A note on two problems in connection with graphs. Numerische Mathematik 1: p269-271
2. G.B. Dantzig., 1963. Linear Programming and Extensions, Princeton University Press, Princeton, NJ. p1-625
3. Kambo N.S., 1984. Mathematical Programming Technioques, East-West Press, p124-131
4. Ralph P. Grimaldi., 2003. Discrete and Combinatorial Mathematics- An Applied Introduction, Pearson fifth Edition. Wesley, p591-625
5. Sohana, J, and Sajid,H,Md. 2011. A Comparitive Study on Algorithms for Shortest-Route Problem and some Extensions. International Journal of Basic and Applied Sciences. Vol.11, No. 6. p167-177
6. Vanderbei Robert J., 2008. Linear Programming Foundations and Extensions, Springer, p55-68

Dr. K.M. Roopa, Professor, Dept. of Mathematics, Bangalore Institute of Technology, Bangalore-560004.

Mis. H. R. Apoorva, Software Developer, Amazon India, Bangalore
Mr. V.K. Srinivasu, Scientific Officer, KSTA, Dept. of Science and Technology, GoK, Bangalore.
Mr. M.C.Viswanath, Assistant Professor, Dept. of Mathematics, Nagarjuna college of Engineering and Technology, Bangalore.