[Volume 2 issue 12 December 2014] Page No.751-765 ISSN :2320-7167

INTERNATIONAL JOURNAL OF MATHEMATICS AND COMPUTER RESEARCH

A Comparative Study on Different Algorithms for finding Longest Route in a Project Network

Dr. K.M. Roopa*, H.R. Apoorva[1] and M.C.Viswanath[2]

*Professor, Department of Mathematics, Bangalore Institute of Technology, Bangalore, India
Email: roopakm10@gmail.com
[1]B.E, Department of Computer Science, PES Institute of Technology, Bangalore, India
[2]Assistant Professor, Department of Mathematics, Nagarjuna College of Engg. & Technology, Bangalore, India

Abstract:

Network techniques are often used in scheduling projects that contain many interrelated activities. One approach that has been widely used is the Critical Path Method(CPM), in which a network diagram depicts precedence among activities. In the CPM, such as three parameters earliest event time, latest event time and slack time are used to determine each of the critical activities. This paper also deals with the methodology to find the critical path using modified Dijkstra's algorithm, Floyd Warshall's algorithm and Bellman Ford algorithm. In addition to critical path problem formulated as a Linear Programming Problem (LPP), it was also solved using binary integer (0-1) programming formulation. Further, dual of the linear programming is used to determine the critical path as well as critical distance. Finally, these algorithms were executed using matlab software.

Keywords: Longest Route, Critical Path Method, Modified Dijkstra's, Floyd Warshall's, Bellman Ford algorithms, LPP, Matlab, Network Analysis

1.0 Introduction:

Scheduling activities in project management is becoming increasingly important to obtain competitive priorities such as on-time delivery. By using project management, managers are able to obtain a graphical display of project activities (tasks) and an estimate of how long the project will take to complete. The Critical Path Method (CPM) is a network-based method, designed to assist in the planning, scheduling and control of the project. Its objective is to construct the time scheduling for the project. Two basic results provided by CPM are the total duration needed to complete the project and the critical path. One of the procedures for finding a critical path is to first determine the earliest occurrence times via a forward pass and the latest occurrence times via a backward pass of each activity and then identify critical activity, an activity with the earliest occurrence time equal to the latest occurrence time [5]. Finally, critical activities constitute a critical path that is a single uninterrupted path spanning the entire project network from start to finish, and consequently the total duration time of this project network can be obtained by summing these critical activity times. However, it is possible for a project network to have more than one critical path [4] [8].

Dijkstra's algorithm computes the shortest paths from a source vertex to every other vertex in a graph [1]. In this paper, modified Dijkstra's algorithm [5] is presented to find critical path and critical distance in a project network. Floyd Warshall's [2, 7] and Bellman Ford algorithms are also discussed to find critical path by inverting edge weight in a directed acyclic graph [6]. Further, these algorithms are executed using matlab software. The linear programming formulation for critical path problem was solved using binary integer (0-1) programming technique, which is also discussed. In addition, CPM is formulated as binary integer programming and dual linear programming problems, which is solved by algebraic method [3]. For large number of vertex, matlab software can be a best choice.

2.0 Critical Path Method (CPM)

A directed acyclic graph with non negative weights and unique source and destination is called an activity network. In this network, the arc represents activity, while the precedence relationship between all activities is represented by the topology of the network. Let G=(V, A, T, s, d) be an activated network, where V={v_1,v_2……v_n} be the set of vertices, in which v_1 and v_n represent the start and final events of the project respectively. Let $A \subseteq V \times V$ be the set of directed edges, a_{ij}= (v_i, v_j), that represents the activities to be performed in the project. The activity a_{ij} is represented by one and only one arrow with tail event v_i and head event v_j. For each activity a_{ij}, a magnitude t_{ij}ЄT is defined, where t_{ij} is the time required for completion of that particular activity. With respect to CPM, the forward pass calculates the earliest time at which the activity can start provided that its precedent activities are completed first and the backward pass calculates the latest time at which the activity can be completed without delaying the project. Float is the amount of time an activity can be delayed or lengthened. This is also called slack. Arcs on the critical path have zero float, while non critical arcs have non zero floats. The more accurate term for float is total float but often shortened as float, since it is, by far, the most commonly used measure. Critical path is the longest path from v_1 to v_n and an activity a_{ij} on critical path is called the critical activity. The steps used to find the critical path are as follows:

i) Perform a forward pass through the network to determine the earliest time say, E and longest distance from v_1 to v_n.

ii) Perform a backward pass through the network to determine the latest time say, L.
iii) Calculate the float for each arc to determine the longest path in the project network.

2.1 Forward pass algorithm:

Step1: Let the start event be v_1 and set earliest time E_1=0. Also, the set S represents ascending sequence of topological order.

Step2: For each event in S, repeat the following steps.

Step2 (a): Calculate the earliest time for the current event as below:

$$E_j=\max\{E_a+t_{aj}, E_b+t_{bj} \ldots\ldots\ldots\ldots\ldots E_k+t_{kj}\} \qquad (1)$$

Where, a, b ……k are the head events of activities from event j.

E_a, E_b………E_k are the earliest time of the events a, b …k respectively.

t_{aj}, t_{bj}………t_{kj} are the corresponding time taken to complete of each activity.

Step2 (b): If the current event is same as the final event then the algorithm stops.

2.2 Backward pass algorithm:

Step1: Let final event be v_n and set latest time L_n=E_n. Also, the set S' represents descending sequence of topological order.

Step2: For each event in S', repeat the following steps.

Step2 (a): Calculate the latest time for the current event as below:

$$L_i=\min\{L_a-t_{aj}, L_b-t_{bj} \ldots\ldots\ldots\ldots\ldots L_k-t_{kj}\} \qquad (2)$$

Where, a, b ……k are the tail events of activities from event i.

L_a, L_b………L_k are the latest time of the events a, b …k respectively.

t_{aj}, t_{bj}………t_{kj} are the corresponding time taken to complete of each activity.

Step2 (b): If the current event is same as the start event then the algorithm stops.

2.3 Total Float:

Total float is used to determine whether an event is critical. If the total float for an event is zero then that event is critical event. The critical path is then found by connecting critical events.

Steps to compute total float:

Step1: Let the start event v_1 be the current event. Also the set S represents the ascending sequence of topological order.

Step2: For each event in S, repeat the following steps.

Step2 (a): If the current event is same as the final event, v_n then the algorithm stops.

Step2 (b): For each event that is the head event of an activity from current event calculate total float as follows

$$T_{ij} = L_j - E_i - t_{ij} \quad (3)$$

Where i= current event, j= successor event

Deduce the critical path:

Step1: Let P be the set of all events that make up the critical path. Add the start event v_1 to P.

Step 2: Start with the T_{1j} value that is zero and include the event v_j in P. From this find a T_{jk} value that is zero and include the event v_k in P. Continue the process until you find a T_{mn} value that is zero.

Step 3: Add the final event v_n to set P.

Example: Consider the activity network as shown in Figure1, determine the earliest time and latest time by forward pass and backward pass algorithm. Also calculates total float.

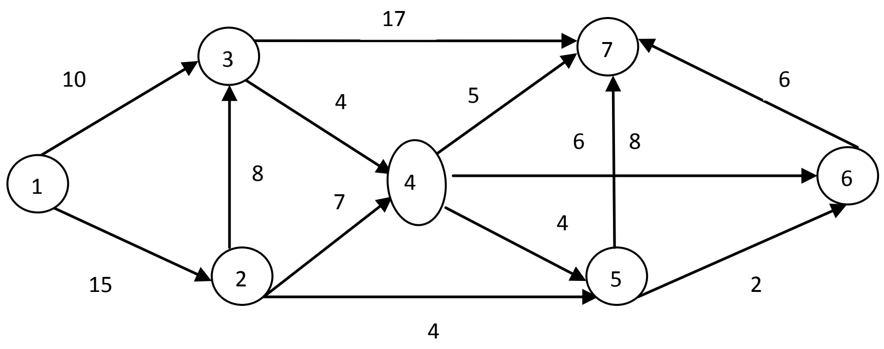


Fig.1: Activity network

Forward pass

Set $E_1=0$: $E_2= E_1+ t_{12}= 0+15= 15$: $E_3= \max \{E_1+ t_{13}, E_2+ t_{23}\} = \max \{10, 23\} =23$

$E_4= \max\{ E_2+t_{24}, E_3+t_{34}\}= \max\{22, 27\}=27$: $E_5= \max\{ E_2+t_{25}, E_4+t_{45}\}= \max\{19, 31\}=31$

$E_6= \max \{E_4+t_{46}, E_5+t_{56}\} = \max \{33, 33\} =33$: $E_7= \max\{ E_3+ t_{37}, E_4+ t_{47}, E_5+ t_{57}, E_6+ t_{67} \}=\max\{40,32,39,39\}=40$

Backward pass

Set $E_7=L_7=40$: $L_6=L_7-t_{67}=40-6=34$: $L_5= \min \{ L_7- t_{57}, L_6- t_{56}\} =\min \{32, 32\} =32$

$L_4= \min\{ L_7- t_{47}, L_6- t_{46}, L_5- t_{45}\}=\min \{35,28,28\}=28$: $L_3= \min\{ L_7- t_{37}, L_4- t_{34}\}=\min \{23,24\}=23$

$L_2= \min\{ L_3- t_{23}, L_4- t_{24}, L_5- t_{25}\}=\min \{15,21,28\}=15$: $L_1= \min\{ L_2- t_{12}, L_3- t_{13}\}=\min \{0,13\}=0$

Total float of the activity

$T_{12}=L_2-E_1-t_{12}=0$: $T_{13}=L_3-E_1-t_{13}=13$: $T_{23}=L_3-E_2-t_{23}=0$: $T_{24}=L_4-E_2-t_{24}=5$: $T_{25}=L_5-E_2-t_{25}=13$

$T_{34}=L_4-E_3-t_{34}=1$: $T_{37}=L_7-E_3-t_{37}=0$: $T_{45}=L_5-E_4-t_{45}=1$: $T_{46}=L_6-E_4-t_{46}=1$: $T_{47}=L_7-E_4-t_{47}=8$

$T_{56}=L_6-E_5 - t_{56}=1$: $T_{57}=L_7-E_5- t_{57}=1$: $T_{67}=L_7-E_6 - t_{67}=1$

Therefore the critical path of the project network is 1-2,2-3,3-7 i.e. 1-2-3-7

3.0 Modified 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. Modified Dijkstra's algorithm is used to find the path with maximum 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 longest distance. Further, the algorithm provides means to trace the longest path (critical path) accordingly.

3.1 Outline of Modified's Algorithm:

Modified 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 $i \neq j$, 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 zero.

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

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

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 $j\neq s$, that are output from the previous iteration are taken as inputs for the current iteration.

Initially P = {s}.where S is the start vertex. S is a set of size n populated with i) 0 if $t_j = 0$, ii) s if t_j = Non-Zero value.

The steps involved in each iteration for finding the longest path are summarized below:

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

Add k to the set P.

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

$$t_j = \max\{ t_j, t_k + q_{kj} \} \qquad (5)$$

If $(t_k + q_{kj}) > t_j$ then update the j_{th} entry in S to k.

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

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

Step 1: Take node e as the last node in the longest path. Add e to the partially constructed critical path.

Step 2: Find the e^{th} entry in the set S, let this be x. Add this prefix node x to the partially constructed critical path.

Step 3: Check whether x is equal to s. If so, go to Step 4; else set e = x and go to Step 2.

Step 4: Add s to the partially constructed critical path. The required critical path from node s to node e is obtained.

Example: Determine the Critical Path and Critical distance from vertex 1 to vertex 7 in the weighted directed network is depicted in the fig.1.

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

$t_2 = 15, t_3 = 10, t_4 = 0, t_5 = 0, t_6 = 0, t_7 = 0$

Step 1: Note that t_j is maximum for j=2, i.e., t_2 =15 .Therefore add 2 to the set P

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

$t_3 = \max \{t_3, t_2 + q_{23}\} \Rightarrow t_3 = \max \{10, 15+ 8\} =23 \Rightarrow S [3] = 2$

$t_4 = \max \{t_4, t_2 + q_{24}\} \Rightarrow t_4 = \max \{0, 15 + 7\} =22 \Rightarrow S [4] = 2$

$t_5 = \max \{t_5, t_2 + q_{25}\} \Rightarrow t_5 = \max \{0, 15+4\} = 19. \Rightarrow S [5] = 2$

Second Iteration: P= {1, 2}, S={0,1,1,2,2,0,0} , and $t_2 = 15, t_3 = 23, t_4 = 22, t_5 = 19, t_6=0, t_7 =0$.

Step 1: For the set of recalculated t_j values, maximum occurs at j=3, i.e.t_3 =23. Therefore, add node 3 to set P.

Step 2: Now, P = {1, 2, 3} and t_3 =23

$t_4 = \max \{22, 23 + 4\} =27 \Rightarrow S [4] = 3$: $t_7 =\max \{0, 23+17\} = 40 \Rightarrow S [7] = 3$

The temporary longest path from vertex1 to vertex7 is 1-2-3-7 which is of cost 40.

Check if there is any other path that is longer other than this path.

Second distance in this case from vertex 4.

Third Iteration: P = {1, 2, 3}, S= {0, 1, 1, 2, 2, 3, 3}, and $t_2 = 15, t_3 = 23, t_4 = 27, t_5 = 19, t_6 = 0, t_7 = 0$.

Step 1: The maximum t_j value occurs for j=4, i.e. t_4 =27. Add node to the set P.

Step 2: P= {1, 2, 3, 4} and t_4 =27

$t_5 = \max \{t_5, t_4 + q_{45}\} \Rightarrow t_5 = \max \{19, 27+ 4\} =31 \Rightarrow S [5]=4$

$t_6 = \max \{t_6, t_4 + q_{46}\} \Rightarrow t_6 = \max \{0, 27+ 6\} =33 \Rightarrow S [6] = 4$

$t_7 = \max \{t_7, t_4 + q_{47}\} \Rightarrow t_7 = \max \{0, 27+ 5\} =32. \Rightarrow S [7] = 4$

The path 1-2-3-4-7 =32 is shorter than then the path 1-2-3-7=40.

Third distance from vertex 5

Fourth Iteration: P={1, 2, 3, 4}, S = {0, 1, 1, 2, 2, 3, 3}, and $t_2 = 15, t_3 = 23, t_4 = 27, t_5 =31, t_6 = 0, t_7 = 0$.

Step 1: The maximum t_j value occurs for j=5, i.e. t_5 =31. Add node 5 to the set P.

Step 2: P= {1, 2, 3, 4, 5 } and t_5 =31

$t_6 = \max \{t_6, t_5 + q_{56}\} \Rightarrow t_6 = \max \{0, 31+ 6\} =33 \Rightarrow S [6] = 5$

$t_7 = \max \{t_7, t_5 + q_{57}\} \Rightarrow t_7 = \max \{0, 31+ 8\} =39. \Rightarrow S [7] = 5$

The path 1-2-3-4-5-7 =39 is shorter than then the path 1-2-3-7=40.

Third distance from vertex 6

Fifth Iteration: P={1, 2, 3, 4, 5}, S ={0, 1, 1, 2, 2, 3, 3}, and $t_2 = 15, t_3 = 23, t_4 = 27, t_5 =31, t_6 = 33, t_7=0$.

Step 1: The maximum t_j value occurs for j=6, i.e. t_6 =33. Add node 6 to the set P.

Step 2: P= {1, 2, 3, 4, 5, 6} and t_6 =33

t_7 = max {t_7, t_6 + q_{67}} => t_7 = max {0, 33+ 6} =39. => S [7] = 6

The path 1-2-3-4-5-6-7 = 39 is shorter than the path 1-2-3-7=40.

Therefore the critical path from vertex1 to vertex7 is 1-2-3-7 which is of duration 40 days.

The Modified Dijkstra's algorithm is implemented in Matlab software and the corresponding output is given below:
dist =40 ; path =1 2 3 7; pred = 1 1 2 3 4 4 3

Therefore, the critical path in the project network is 1→2→3→7 with the duration of 40 days.

4.0 Floyd Warshall algorithm and Bellman Ford algorithm:

These algorithms are basically used to solve shortest path problems with positive or negative edge weight but without negative cycle. However, it is possible to find longest path by taking negative sign of their edge weight. This transformation cannot be used in most of the graphs, because it creates a cycle of negative length in graph G. As the graph to find critical path is a directed acyclic graph with positive edges, it does not create any negative cycles. Further, these algorithms can also be implemented using matlab software.

4.1 Floyd Warshall algorithm:

This algorithm has square matrix d, with n-rows and n- columns. Each entry d (i, j) of the matrix gives the critical path from node i to node j. If there is a direct link from node i to node j, it is finite or else it is infinite i.e. d(i, j) = ∞. Floyd Warshall's Algorithm is based on transitivity property. Consider the d (i, j) three nodes i, j and k as shown in Figure 2.0. From the transitivity property,

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

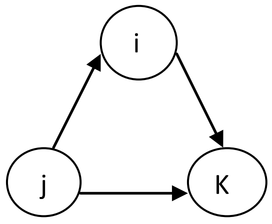


Fig. 2.0

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), it is optimal to replace the direct route from i→k, with the indirect route, being i→j→k.

4.2 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 to the current iteration and the n^{th} iteration where n is the number of nodes in the graph that gives the optimal/final distance matrix D as well as the final precedence matrix S. The optimal distance matrix D_n represents the shortest distances between any two nodes in the network. However, by taking negativethe sign of the edges and then applying Floyd Warshall's Algorithm, the distance matrix D_n represents the longest distance between any two nodes in the network and the corresponding critical path can be traced out from the precedence matrix, S_n. The steps for finding critical path under Floyd Warshall's 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/column 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) if D_{k-1} with d (i, k) + d (k, j)

(ii) Create S_k by replacing S (i, j) in S_{k-1} with k. set k=k+1. If k > n, the algorithm stops else repeat steps between 2 and 4. Reverting the sign of the entry $D_n(s, n)$ gives the critical distance.

Case (b): If d(i, k) + d(k, j) = d(i, j), (i ≠ k, j ≠ k and i ≠ j), do not make any change this implies that i → k → j is an alternative way of i → j.

Similarly, steps to trace the critical path between two nodes, says i and j using Floyd Warshall's algorithm are given below

Step1: Take node n as the last node in the shortest path.

Step2: Find the S (1, n) entry in the matrix S_n, let this be x. add this prefix node x to the partially constructed critical path.

Step3: Check S (1, x) =x. If so, go to step 4; else set n=x and go to step 2.

Step4: Add S to the partially constructed critical path. The required critical path from node S to n is obtained. On reverting the sign of the edge weight, the shortest path now becomes the critical path.

Example: Consider the problem of determining critical path in the project network as shown in Figure 2.1

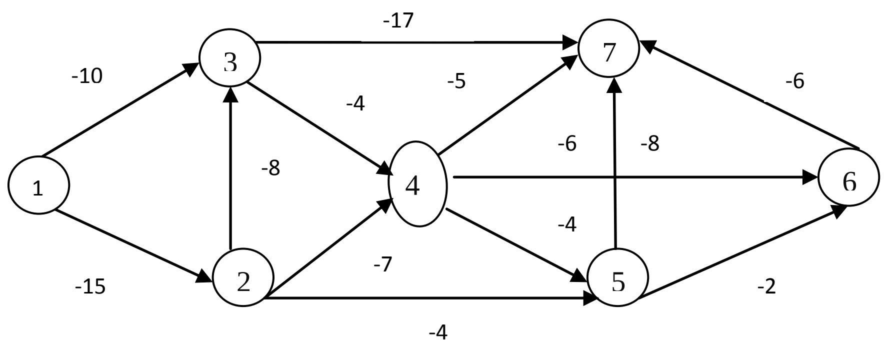


Fig. 2.1

Iteration (0): Consider the initial representation of the matrix D_0 and S_0 from the 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	∞	-	-8	-7	-4	∞	∞
3	∞	∞	-	-4	∞	∞	-17
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, by considering first column and first row as pivot column and pivot row in the distance matrix D_0, all the entries of the pivot column in D_0 are infinity. Hence, D_1 and S_1 are same as from D_0 and S_0.

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

3	∞	∞	-	-4	∞	∞	-17
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,by considering second column and second row as pivot column and pivot row in the matrix D_1 , except d(1,2) all the entries in the pivot column are infinity. Also except d (2, 3), d (2,4) and d(2,5), all the entries in the pivot row are infinity. Now apply transitivity property to obtain the following results:

(i) d(1,2)+d(2,3)= d(1,3)
But, d(1,2)+d(2,3)=-15 - 10 and d(1,3) =-10

This implies -23< -10, then d (1, 3) =-23

d(1,2)+d(2,4)=d(1,4)
But, d(1,2)+ d(2,4)= -15 - 7 and d(1,4)=∞

this implies -22<∞, t hen d (1, 4) = -22

Similarly -19<∞ , then d (1, 5) = -19.

(ii) The precedence matrix S_1 can be changed as

S (1, 3) = 2, S (1, 4) = 2 & S (1, 5) = 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	-23	-22	-19	∞	∞
2	∞	-	-8	-7	-4	∞	∞
3	∞	∞	-	-4	∞	∞	-17
4	∞	∞	∞	-	-4	-6	-5
5	∞	∞	∞	∞	-	-2	-8
6	∞	∞	∞	∞	∞	-	-6
7	∞	∞	∞	∞	∞	∞	-

S_2	1	2	3	4	5	6	7
1	1	2	2	2	2	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 (3): Set k=3, by considering third column and third row as pivot column and pivot row in the matrix D_2, except d(1,3) and d(2,3) all the entries in the pivot column and also except d(3,4) and d(3,7) in the pivot row are infinity. Further, apply transitivity property to obtain the following results:

(i) Since, d(1,4)= -27 , d(1,7)=-40 and d(2,4)=-12 . So, d (2,7) = -25

(ii) Set precedence matrix S_2 as S (1, 4) = 3, S (1, 7) = 3, S (2, 4) = 3, S (2, 7) = 3 These are the changes as shown in the matrix D_3 and S_3.

D_3	1	2	3	4	5	6	7
1	-	-15	-23	-27	-19	∞	-40
2	∞	-	-8	-12	-4	∞	-25
3	∞	∞	-	-4	∞	∞	-17
4	∞	∞	∞	-	-4	-6	-5
5	∞	∞	∞	∞	-	-2	-8
6	∞	∞	∞	∞	∞	-	-6
7	∞	∞	∞	∞	∞	∞	-

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

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	-23	-27	-31	-33	-40

2	∞	-	-8	-12	-16	-18	-25
3	∞	∞	-	-4	-8	-10	-17
4	∞	∞	∞	-	-4	-6	-12
5	∞	∞	∞	∞	-	-2	-8
6	∞	∞	∞	∞	∞	-	-6
7	∞	∞	∞	∞	∞	∞	-

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

The minimum weighted path 1 to 7 is the critical path and inverting the sign of the weight of this path gives the critical distance. Now, it is found that the longest distance between any two nodes can be determined from the matrix D_7 and also the route can be determined from the matrix S_7. For example, the longest path from node 1 to node 7 is 40. Let critical path be the set of all nodes that make up the critical path. Node 7 is the last node in the critical path. Critical path= {7}, S (1, 7) =3, However S (1, 3) = 2≠3. Critical path = {7, 3}, S (1, 7) =2 and also S (1, 2) =2. Critical Path = {7, 3, 2}. Node 1 is the start node of the critical path, critical path = {7, 3, 2, 1). Therefore 1→2→3→7 is the critical path.

The Floyd Warshall's algorithm can also be implemented in Matlab software and the corresponding output is given below:

```
A=inf(7,7);     A(1,2)= -15; A(1,3)= -10;    A(2,4)= -7; A(2,5)= -4; A(2,3)= -8;
A(3,4)= -4; A(3,7)= -17;    A(4,5)= -4; A(4,6)= -6; A(4,7)= -5;   A(5,6)= -2; A(5,7)= -8;  A(6,7)= -6;

[Q, P, result]=FloydSPR (A, 1, 7)
Q =                                      |  P =
   Inf  -15   -23  -27   -31   -33 -40   |     -1   -1    2    2    2    2    2
   Inf  Inf    -8  -12   -16   -18  -25  |     -1   -1   -1    3    3    3    3
   Inf  Inf   Inf   -4    -8   -10  -17  |     -1   -1   -1   -1    4    4   -1
   Inf  Inf   Inf  Inf    -4    -6  -12  |     -1   -1   -1   -1   -1   -1    5
   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 = - 40. Inverting the sign of this result gives the critical distance.Therefore, the critical path in the project network is 1→2→3→7 with the duration of 40 days.

5.0 Bellman Ford algorithm:

Bellman ford algorithm is a label connecting algorithm that computes single-source shortest route in a weighted graph in which some of the edge weights may be negative. Every route in a weighted graph has a corresponding route weight, which is the sum of the weights of the route edges. In a weighted, directed graph, G= (V, E) with source S and weight function W=E→R, the algorithm returns a Boolean value indicating the presence of negative cycle .The algorithm uses relaxation, reducing an estimate d (v) on the weight of a longest path from the source *S* to each event v€ V until it obtains the actual shortest path weight δ(S, v). The algorithm returns TRUE if and only when the graph contains no negative weight cycles that are reachable from the source. If there is negative cycle then the algorithm in turn the critical path.The Bellman Ford algorithm is executed in the following steps.

Step1: Assign infinity to all nodes other than the source node. The source node 1 is set as zero as shown in Figure 3.0

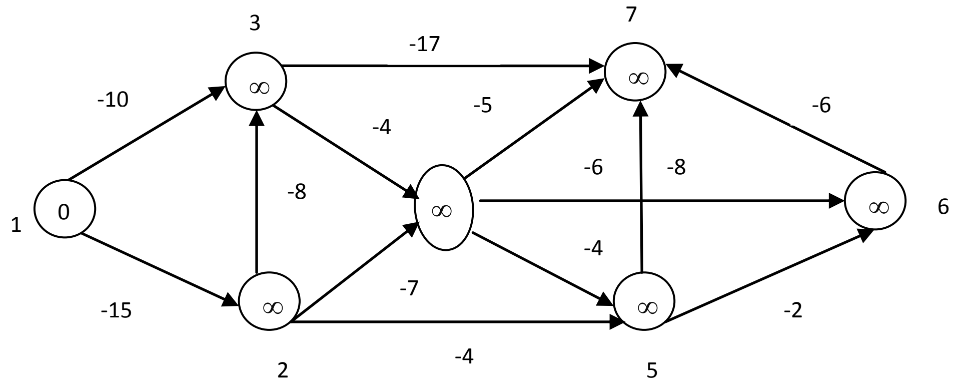


Fig. 3.0

Step2: Relax each edge for (n − 1) times where n are the number of nodes. So relax edges for 6 times since n is 7. Relaxing an edge means verifying if the path to a node to which the edge is pointing can be shortened, and if so, replace that path with the found path. Start by first relaxing the edge: with only 2 nodes, starting from the source node. Consider the edge 1→2 of cost -15, the cost of the source node plus the cost of the edge 1→2 is less than infinity means, which replaces the node 2 with the new cost. Similarly relax edge 1→3 of cost -10 which is also less than infinity. It is shown in Figure 3.1

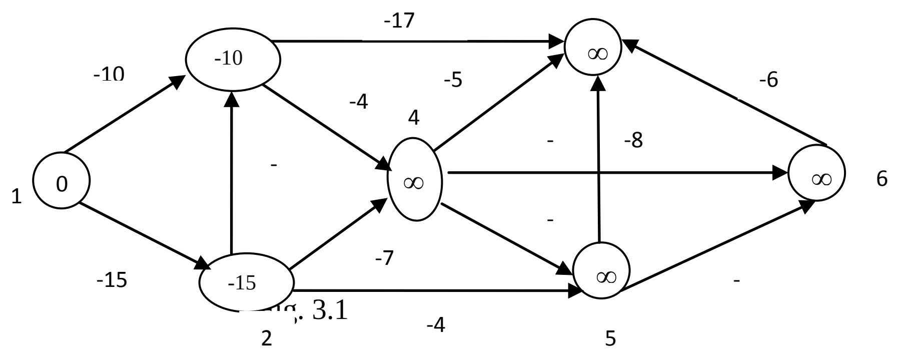


Fig. 3.1

Step3: Relax a path with 3 nodes, starting from source node. With respect to this relax edge 1→3 through 1→2→3=> -23< -10, relax edge 1→4 through 1→2→4=> -22< ∞ , relax edge 1→5 through 1→2→5=> -19<∞ and relax edge 1→7 through 1→3→7=> -27< ∞. The execution of this step is shown in Figure 3.2

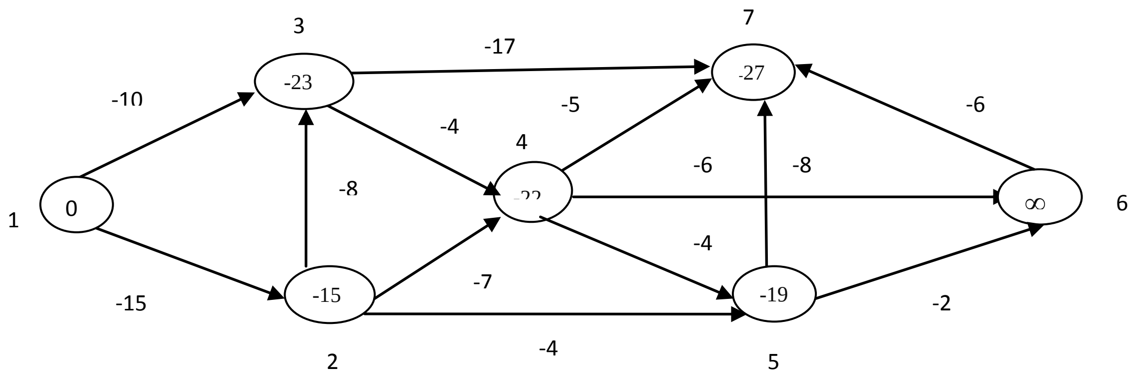


Fig.3.2

Step4: Relax a path with 4 nodes, starting from source node. With respect to this relax edge 1→4 through 1→2→3→4 => -27< -22 and relax edge 1→6 through 1→2→5→6=>-21<∞, relax edge 1→7 through 1→2→3→7=> -40< -27. The network is changed as shown in Figure 3.3

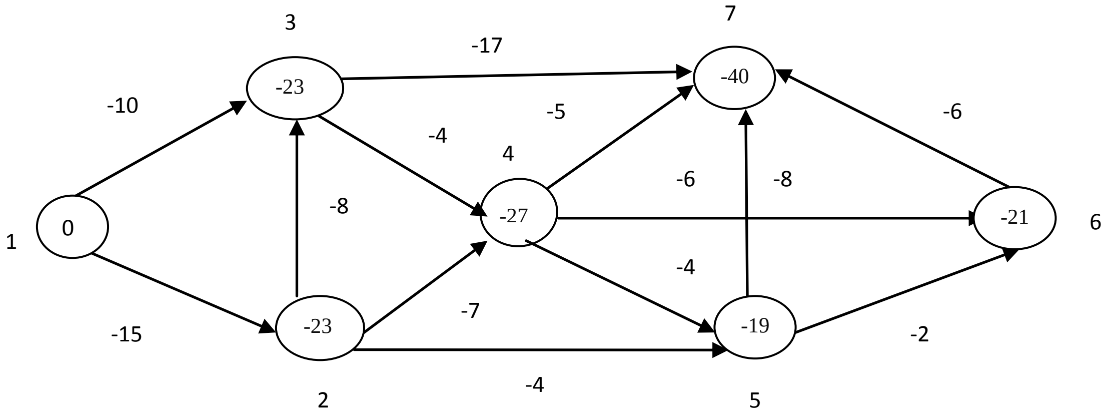


Fig. 3.3

Step5: Relax a path with 5 nodes, starting from source node. With respect to this relax edge 1→5 through 1→2→3→4→5=>-31<-19 and relax edge 1→6 through 1→2→3→4→6=>-33< -21. The execution of this step is shown in Figure 3.4

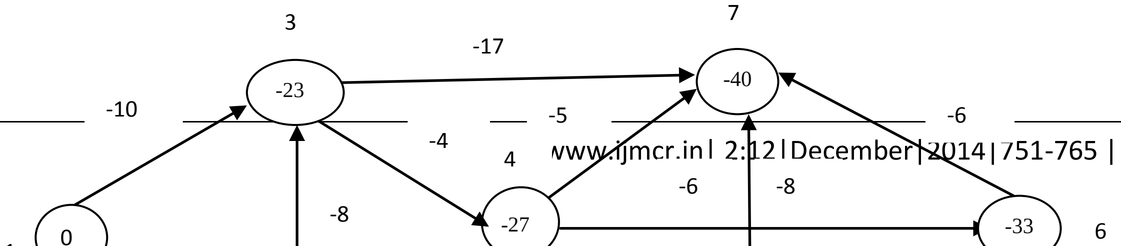


Fig. 3.4

Step 6: Relax a path with 6 nodes, starting from source node. However, all the edges are relaxed and there is no change. The minimum weighted path 1 to 7 is the critical path and inverting the sign of the weight of this path gives the critical distance. The Bellman ford algorithm can be implemented using Matlab software and the corresponding output is given below: duration = -40 ; path =1 2 3 7; pred = 0 1 2 3 4 4 3. Therefore the critical path in the project network is 1→2→3→7 with the longest duration of 40 days.

6.0 Linear programming formulation of the longest path problem:

It is a very simple programming formulation problem. Most of the network problems can be formulated as linear programming problems and can be solved by simplex method algorithm. In this section, two linear programming formulations for the longest-route (critical path) problem are discussed. These formulations are generally used to find the critical path between any two nodes in the network. The following binary integer programming and dual linear programming formulations can be used to determine the critical path between any two nodes like s and t in the network.

6.1 Binary integer Formulation 1: This formulation assumes that an external one unit of flow enters the network at node s and leaves at node t, where s and t are the two target nodes in which between these nodes to determine the critical path.
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 one time, however, the variable x_{ij} must be assumed that the binary values (0 or 1) only. Thus, the objective function of the linear program becomes
Maximize $Z = \sum_{\text{all defined arcs}} c_{i,j}\ x_{i,j}$,

The given constraint represents the conservation of flow at each node. For any node j will be
Total input flow = Total output flow.

6.2 Dual linear Formulation 2:

This formulation is the dual problem of the linear Formulation1. As the number of constraints in Formulation1 is equal to the number of nodes, the dual problem will have as many variables as the number of nodes in the network. Also, all the dual variables must be unrestricted because 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:

Minimize $Z=y_t - y_s$

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

These formulations problem is also solved by using algebraic method with only small number of variables. Otherwise, this method is very complicated. Also, the matlab software is used to solve these problems.

Example: Consider the problem of determining critical path in the project network as shown in above Figure1

By setting $x_{ij} = \begin{cases} -1 \text{ for node i} \\ 1 \text{ for node j} \end{cases}$ and the corresponding values of Linear program are listed below:

Min Z	X_{12}	X_{13}	X_{23}	X_{24}	X_{25}	X_{34}	X_{37}	X_{45}	X_{46}	X_{47}	X_{56}	X_{57}	X_{67}	
	15	10	8	7	4	4	17	4	6	5	2	8	6	-1
Node1	-1	-1												0

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:

Max Z = $15x_{12}+10x_{13}+8x_{23}+7x_{24}+4x_{25}+4x_{34}+17x_{37}+4x_{45}+6x_{46}+5x_{47}+2x_{56}+8x_{57}+6x_{67}$

Subject to $x_{12}+x_{13} = 1$

$x_{12}-x_{24}-x_{27}+x_{32} -=0$

$x_{13}-x_{32}-x_{34}-x_{35}=0$

$x_{24}+x_{34}-x_{\backslash 45}-x_{46}-x_{47} =0$

$x_{35}+x_{45}-x_{\backslash 56}-x_{57} =0$

$x_{46}+x_{56}-x_{67} =0$

$x_{27}+x_{47}+x_{\backslash 57}+x_{67}=1$, $0\leq x_{ij}\leq 1$, for i ,j=1,2,3,4,5,6,7.

This binary integer programming problem in an algebraic method was solved by introducing slack variables. The first tableau and final tableau of algebraic method are given below:

First Tableau:

NZV	x_{12}	x_{13}	x_{23}	x_{24}	x_{25}	x_{34}	x_{37}	x_{45}	x_{46}	x_{47}	x_{56}	x_{57}	x_{67}	S_1	S_2	S_3	S_4	S_5	S_6	S_7	Qty
S_1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
S_2	1	0	-1	-1	-1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
S_3	0	1	1	0	0	-1	-1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
S_4	0	0	0	1	0	1	0	-1	-1	-1	0	0	0	0	0	0	1	0	0	0	0
S_5	0	0	0	0	1	0	0	1	0	0	-1	-1	0	0	0	0	0	1	0	0	0
S_6	0	0	0	0	0	0	0	0	1	0	1	0	-1	0	0	0	0	0	1	0	0
S_7	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0	0	0	0	0	1	1
Δ	-15	-10	-8	-7	-4	-4	-17	-4	-6	-5	-2	-8	-6	0	0	0	0	0	0	0	0

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

Final Tableau:

NZV	x_{12}	x_{13}	x_{23}	x_{24}	x_{25}	x_{34}	x_{37}	x_{45}	x_{46}	x_{47}	x_{56}	x_{57}	x_{67}	S_1	S_2	S_3	S_4	S_5	S_6	S_7	Qty
x_{23}	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
x_{12}	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
S_3	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	1	1	1	1	1	0
S_4	0	0	0	1	1	1	0	0	0	-1	0	-1	-1	0	0	0	1	1	1	0	0
S_5	0	0	0	0	1	0	0	1	0	0	-1	-1	0	0	0	0	0	1	0	0	0
S_6	0	0	0	0	0	0	0	0	1	0	1	0	-1	0	0	0	0	0	1	0	0
x_{37}	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0	0	0	0	0	1	1
Δ	0	13	0	5	12	0	0	0	0	8	0	1	1	15	0	8	12	16	18	25	40

From the final tableau, the optimal solution was found to be Z=40 at $x_{12}=1$, $x_{23}=1$, $x_{37}=1$. This solution gives the critical path from node 1 to node 7 as $1\rightarrow 2\rightarrow 3\rightarrow 7$ and the associated duration of z = 40 days.

It can also be solved from the above binary integer programming problem using Matlab software and the corresponding solutions are given below:
$Z=40$ at $x_{12}=1$, $x_{23}=1$, $x_{37}=1$.
This solution gives the critical path from node 1 to node 7 as $1\rightarrow 2\rightarrow 3\rightarrow 7$ and the associated duration of $z = 40$ days.

From the concept of the dual linear programming problem, the following objective function and constraints are obtained:

Minimize $Z = y_7 - y_1$

Subject to

$y_2 - y_1 \geq 15$ (Route 1 to 2), $y_3 - y_1 \geq 10$ (Route 1 to 3), $y_3 - y_2 \geq 8$ (Route 2 to3), $y_4 - y_2 \geq 7$ (Route 2 to 4),
$y_4 - y_3 \geq 4$ (Route 3 to 4), $y_7 - y_3 \geq 17$ (Route 3 to 7), $y_5 - y_2 \geq 4$ (Route 2 to5), $y_5 - y_4 \geq 4$ (Route 4 to 5),
$y_6 - y_4 \geq 6$ (Route 4 to 6), $y_7 - y_4 \geq 5$ (Route 4 to 7), $y_6 - y_5 \geq 2$ (Route 5 to 6), $y_7 - y_6 \geq 6$ (Route 6 to 7),
$y_7 - y_5 \geq 8$ (Route 5 to7). (6)

Where, $y_1, y_2, \ldots\ldots y_7$ are unrestricted sign.
This linear programming problem in an algebraic method was solved by introducing surplus variables. The first tableau and final tableau of algebraic method are given below:

First Tableau:

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

Final Tableau:

NZV	y_7	y_6	y_5	y_4	y_3	y_2	y_1	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
y_2	0	0	0	0	0	1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	15
y_3	0	0	0	0	1	0	-1	-1	0	-1	0	0	0	0	0	0	0	0	0	0	23
S_3	0	0	0	0	0	0	0	-1	1	-1	0	0	0	0	0	0	0	0	0	0	13
y_4	0	0	0	1	0	0	-1	-1	0	-1	0	0	-1	0	0	1	0	0	1	0	28
S_5	0	0	0	0	0	0	0	0	0	-1	1	0	-1	0	0	1	0	0	1	0	6
S_6	0	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	1	0	0	1	0	1
y_5	0	0	1	0	0	0	-1	-1	0	-1	0	0	-1	0	0	0	0	0	0	1	32

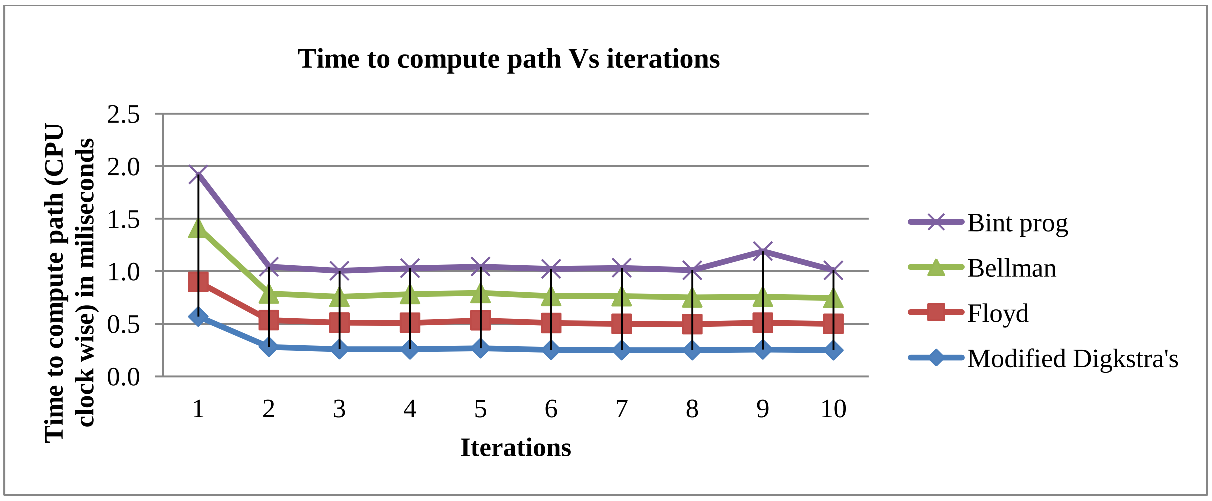
S_8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	-1	1	0
S_9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	-1	0	7
S_{10}	0	0	0	0	0	0	0	0	0	-1	0	0	-1	1	0	0	0	0	0	1	13
S_{11}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	-1	0
y_7	1	0	0	0	0	0	-1	-1	0	-1	0	0	-1	0	0	0	0	0	0	0	40
y_6	0	1	0	0	0	0	-1	-1	0	-1	0	0	-1	0	0	0	0	0	1	0	34
Δ	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	-40

From the final tableau, the optimal solution of Z=40 at y_1 =0, y_2 =15, y_3 =23, y_4 =28, y_5 =32, y_6 =34, y_7 =40,

These solutions have satisfied the constraints (6) .Therefore, it is clear that the value of Z = 40 gives the critical path from node 1 to node 7 and the longest route 1-2, 2-3, 3-7, 4-5, 4-6 , 5-6, 5-7, 6 -7 are obtained. From these sequence 1-2, 2-3, 3-7 give the critical path 1→2→3→7

It can also be solved the above dual problem using Matlab software and the corresponding solutions are: Z=40 at y_1 = 8.3278, y_2 = - 6.6722, y_3 = -14.6722, y_4 = -18.8366, y_5 = -23.0222, y_6 = -25.4382, y_7 = -31.6722

7.0 Conclusions:



Comparison of computational time of various algorithms

From the above graph, it can be concluded that the modified Dijkstra's is the most effective algorithm to find the longest route in a large network when compared to other most discussed algorithms such as Floyd's Warshall's algorithm and Bellman Ford algorithm as well as some of the analytic methods like LPP, Dual LPP and algebraic method. This study has also revealed that some of the algorithms and analytic methods demonstrated in matlab software could be the ideal choice for determining the longest path in larger networks. Therefore, the proposed methods can be easily applied to identify the longest path in larger project networks like health care, water and power distribution, plant construction and building construction etc.

Acknowledgement:

The support got from Dr.A.M. Ramesh, Senior Scientific Officer and Srinivasu V.K, Scientific Officer from Karnataka Science and Technology Academy is gratefully acknowledged.

References:

1. Ahuja H.N, Dozzi S.P, and Abourizk S.M,1994. Project Management, New York: Wiley.
2. Dijkstra, E. W., 1959. A note on two problems in connection with graphs. Numerische Mathematik 1: p269-271.
3. Dr. Roopa K.M, Apoorva H.R., Srinivasu V.K. and Vishwanath M.C, 2013. A Study on Different Algorithm for shortest Route Problem. IJERT, 2(9): P422-434.

4. Hemalatha S and Valsalal P, 2012. Identification of Optimal Path in Power System Network Using Bellman Ford Algorithm. Hindawi Publishing Corporation, Modelling and Simulation in Engineering,Article ID 913485.

5. Ravi Shankar N and Sireesha, 2010. Using modified Dijkstra's Algorithm for Critical Path Method in a Project Network. IJCAM, 5 (2) : p.217-225.

6. 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. 11(6): p167-177

7. Stevenson W.J, 2002. Operation Management, seventh edition, McGraw-Hill.

8. Taha H A, 2003. Operations Research: An Introduction, (Seventh Edition), Prentice Hall.