SINET: Ethiop. J. Sci., 21(2):171–182, 1998

ISSN: 0379–2897

RANKED K-LONGEST PATHS IN AN ACYCLIC NETWORK, ITS ALGORITHM AND APPLICATION

Berhanu Guta

Department of Mathematics, Faculty of Science, Addis Ababa University
PO Box 1176, Addis Ababa, Ethiopia

ABSTRACT: The acyclic network on which our problem is defined is a weighted directed graph $G=(N, A)$ with no directed cycle. A method to determine the first, second, third, .., and k-th longest paths for a given integer $k \geq 2$ is described. An algorithm of $0(k^2m)$ to determine the k-longest path in the network consisting of m arcs is also given. There can be various advantages of determining k-longest path in many industrial, engineering and management problems that deal with planning and scheduling of activities involving n specified jobs subjected to precedence constraints. Indeed, such problems can be modelled mathematically by acyclic networks.

Key words/phrases: Acyclic network, algorithm, CPM, k-longest paths, longest path

INTRODUCTION

We consider a finite directed graph $G=(N, A)$, where N is a set of finite nodes and A is a set of directed arcs in which each arc is joining a pair of nodes. We denote an arc emanated from a node i and incident into a node j by an ordered pair (i, j).

A path P in the directed graph from a node i_1 to a node i_n is a sequence of nodes and arcs alternately occurring in the directed graph, i_1, (i_1, i_2), i_2, (i_2, i_3), i_3, ..., (i_{n-1}, i_n), i_n. We denote the path P by $i_1 - i_2 -,\ldots, -i_{n-1}- i_n$ (Turner, 1970). A directed cycle is a path $i_1 - i_2 -\ldots- i_{n-1}- i_n -i_1$. If a path P contains an arc (i, j), we may describe it as $(i, j) \in$ P.

A weighted directed graph or a network is a directed graph in which a numerical value is associated with its arcs and/or nodes. In the sequel, the weight is a numerical value that is associated with the arcs. The weight of an arc (i, j) may be called the length of the arc and denoted by l_{ij}.

Given a path in a weighted directed graph, the weight of the path is the sum of the weights of all arcs on the path (Hamacher and Queyranne, 1985; Brucker, 1995). The weight of a path P may also be called the length of the path and denoted by l(P), i.e.,

$$l(P) = \Sigma l_{ij}\ , \qquad (i,j) \in P$$

A graph is called an acyclic network if it is a weighted directed graph with no directed cycle (Murty, 1992). Many industrial problems such as activity scheduling problems which involve n jobs with precedence constraints and also many engineering and management problems such as project planning and scheduling can be modelled mathematically by acyclic networks. One of the primary objectives of such planning and scheduling problems is to find a schedule of activities so as to complete the jobs in least possible time. Mathematically, this is the problem of determining the longest weighted path in the acyclic network since the weight of the longest path is just the least possible time for the completion of jobs (Brucker, 1995). It is well known that this problem is solved by the Critical Path Method (CPM).

Even though the technique of CPM and its algorithm can be used to find the longest path in the acyclic network, it never tells us which one is the next longest path, next to the actual longest path. In general, however, it may be desired for a number of reasons to be able to determine, in addition to the longest path, the next or second, third, ..., k-th longest path for an integer $k \geq 2$. Therefore this is to extend the already existing longest path algorithm or CPM to an algorithm that can find these desired paths in the acyclic network.

RANKED k-LONGEST PATHS

Definition

Let P be the set of all paths from the source node to a node j in the acyclic network and let $l: P \rightarrow [0, \infty)$ be the weight function which assigns the length (or distance value) to a given path. Let an integer $k \geq 2$ be given. Then,

1. A set of ranked k-longest paths $R = \{P_1, P_2, P_3, \ldots, P_k\}$ to the node j is the set of paths such that:
   $l(P_1) \geq l(P_2) \geq l(P_3) \geq \ldots \geq l(P_k) \geq l(P) \quad \forall P \notin R.$

2. P is the next longest path to P_r, for $r = 1,2, \ldots, k-1$, if
   - i. both P and P_r are paths from the source node to the node j,
   - ii. $l(P_r) \geq l(P)$, and
   - iii. $l(P_r) \geq l(\tilde{p}) \geq l(P)$ for some path $\tilde{p}$ from the source node to j implies either $l(P_r) = l(\tilde{p})$ or $l(\tilde{p}) = l(P)$.

Assumptions

1. The network has only one source node and only one sink (terminal) node.
2. The network is topologically ordered, i.e., we numbered (or ordered) nodes in the network in such a way $i < j$ for each arc (i, j) in the network.
3. $0 \leq l_{ij} < \infty$, where l_{ij} is the length of an arc (i, j).

Notation

Let P_1^j be actual longest path from source node to the node j. Then,

a. P_{r+1}^j denotes the next longest path to P_r^j, for each $r=1, 2, \ldots, k-1$.

b. At the sink node n, P_r^n is the r-th longest path in the network and may be denoted simply by P_r.

Definition

P_r^j, $r=1, 2\ldots, k-1$, is called the r-th longest path from the source node to a node j in the network.

To determine the set of ranked k-longest paths, we select first a desired integer $k \geq 2$. Then starting from node 1 (the source node), step by step, we find a set of ranked k-longest paths P_1^j, P_2^j, ..., P_k^j to each node j in the network. In general, to each node j we will assign appropriately a k-vector of distance label $d(j) = (d_{j1}, d_{j2}, \dots d_{jk})$ where the r-th component label $d_{jr} = l(P_r^j)$ and $d_{j1} \geq d_{j2} \geq \dots \geq d_{jk}$.

To do this, one can proceed essentially as follows:

Initially set $d_{11} = 0$, i.e., $l(P_1^1) = 0$. This means the first longest path from the source node to itself is assigned 0.

Then set $d_{1r} = -\infty$, i.e., $l(P_r^1) = -\infty$, for r=2, 3,..., k, which means there are no other second, third, ..., or k-paths from the source node to itself.

Thus we have, at the beginning, $d(1) = (0, -\infty, \dots, -\infty)$. Then for any other node j = 2, 3, ..., n initially we determine, of all paths to j, a longest path from node 1 to node j and its distance using the forward path algorithm of the CPM (Murty, 1992, p. 413). This is the first longest path to j, $l(P_1^j)$. We therefore set $d_{j1} = l(P_1^j)$. Once we get, P_1^j we cut (or exclude) this path temporarily, i.e., exclude all arcs on P_1^j, and find the next longest path to the node j. Cutting this path is necessary, otherwise it reappears in the next steps. To cut this path, numerically we assign temporarily $l(P_1^j) := -\infty$, i.e., a sufficiently large negative number, say -99999, which depends on the particular problem.

After cutting P_1^j, we determine a longest path to j and its distance from all remaining paths to j. This path is P_2^j, the second longest path to node j. Thus we assign $d_{j2} = l(P_2^j)$ and cut also this path in the same way as P_1^j is cut. We continue in this way until P_k^j and d_{jk} are determined. If there is no r-th path to a node j, we put $d_{jr} = -\infty$.

In each step, the cutting of a path should be temporary since we may need the path later when we find labels of nodes j+1, j+2, ..., n, since the r-th longest path to j may share some arcs with the next one. The key to the procedure is, in fact, the ability to cut a path temporarily. We will see a way to do this in the next algorithm.

In the next algorithm for each node j we need information about its predecessor node corresponding to the r-th longest path to j. For this purpose we have to construct a predecessor index pred(j(r)) which records a predecessor of the node j that lies on P_r^j. In such cases, if pred(j(r))=i(r_0), for some $r_0 \in \{1,2, ..., k\}$, then this is to mean node i is the predecessor of the node j that lies on P_r^j and, going back, we determine the predecessor of the node that lies on P_r^j from pred(i(r_0)); and so on. At the termination of the algorithm, we get the r-th longest path P_r by the help of the predecessor indices starting from the sink node n, following pred(n(r)) back to the node 1.

For each node j, the precedence node(s) can be viewed as a k-vector

$$(\text{pred }(j(1)), \text{pred }(j(2)), ..., \text{pred}(j(k))) = (i_1(r_1), i_2(r_1), ..., i_k(r_k)),$$

whenever there are k distinct paths to j. Observe that, however, pred (j(r)) can be i(r_0) only for some r_0 such that $1 \leq r_0 \leq r$ and $i \in B(j) = \{i/(i,j) \in A\}$.

RANKED K-LONGEST PATH ALGORITHM

Consider a topologically ordered acyclic network, and a given integer $k \geq 2$. In the algorithm, the variable Temp (i(r)) holds temporarily the length of the r-th longest path to the node i.

Step 1: Initialisation:

Set $j := 1$;
$d_{11} := 0$; $d_{1r} := -\infty$, for $r = 2, 3, ..., k$;

Step 2: Main step:

```
Let $j := j+1$;
Let $B(j) = \{i/(i,j) \in A\}$.
Let Temp $(i(r)) = d_{ir}$ for all $i \in B(j)$ and $r = 1, 2, \ldots, k$;

For $r = 1$ to k Do
Begin
If $r = 1$ then,
 Begin
   $d_{j1} = \max._{i \in B(j)}\{\text{Temp}(i(1)) + l_{ij}\}$;
        Let $i_m$ be any one at which the maximum occurs;
        pred $(j(1)) = i_m(1)$;
        set Temp $(i_m(1)) = -\infty$; (this cuts the path $P_1^j$ temporarily)
End-if

Else
   Begin
        $d_{jr} = \max._{i \in B(j)}\{\text{Temp}(i(t)) + l_{ij}/t = 1, 2, \ldots, r\}$;
                Let $i_m$ $(r_0)$, for $i_m \in B(j)$ & $r_0 \in \{1, 2, \ldots, r\}$, be any one at
                which a positive maximum occur;
                pred $(j(r)) = i_m$ $(r_0)$;
                Set Temp $(i_m(r_0)) = -\infty$;
End-Else
End-For

   $d(j) := (d_{j1}, d_{j2}, \ldots, d_{jk})$;
If $j \leq n$, GOTO STEP 2;

  Else
   For $r = 1$ to k
   $P_r = 1\text{-}i_1\text{-}\ldots\text{-}i_{s-1}\text{-}i_s\text{-}n$, where
        pred $(n(r)) = i_s(.)$, pred $(i_s(.)) = i_{s-1}(.)$, ..., pred $(i_1(.)) = 1(1)$;
STOP.
```

As an example, let us find the ranked 3-longest paths of the following network using the k-longest path algorithm (Fig. 1).

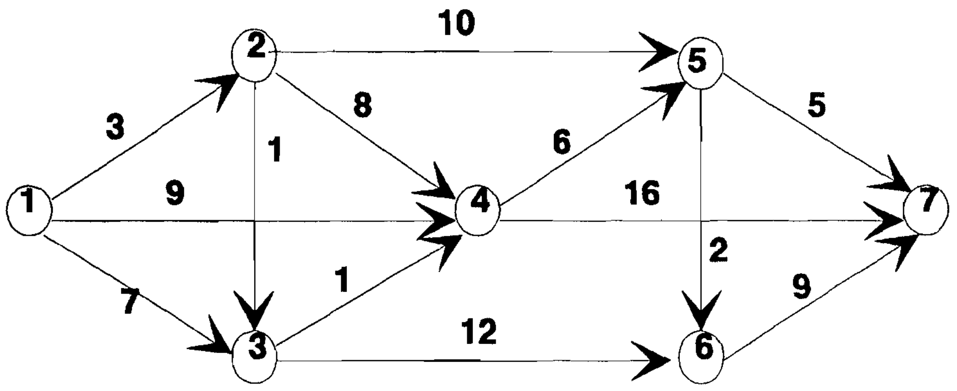

Fig. 1. Sample of an acyclic network. The number on each arc (arrow) represents the weight of the corresponding arc.

Applying the Algorithm to this network with k=3, we can get the following results (Table 1).

Table 1. Distance label and precedence node(s) for each node in the network of Figure 1.

d(1)=(0, -∞, -∞)			
d(2)=(3, -∞, -∞),	pred(2(1))=1(1)		
d(3)=(7, 4, -∞),	pred(3(1))=1(1)	pred(3(2))=2(1)	
d(4)=(11, 9, 8),	pred(4(1))=2(1)	pred(4(2))=1(1)	pred(4(3)=3(1)
d(5)=(17, 15, 14),	pred(5(1))=4(1)	pred(5(2))=4(2)	pred(5(3))=4(3)
d(6)=(19, 19, 17),	pred(6(1))=3(1)	pred(6(2))=5(1)	pred(6(3))=5(2)
d(7)=(28, 28, 27),	pred(7(1))=6(1)	pred(7(2))=6(2)	pred(7(3))=4(1)

Then, we can get the ranked 3-longest paths P_1, P_2, and P_3, of the network using the predecessor indices (Table 1), starting from the sink node 7, following pred(7(r)), for r=1, 2, 3, back to the node 1.

That is,
P_1: 7, pred(7(1))=6(1), pred(6(1))=3(1), pred(3(1))=1(1).

$\Rightarrow P_1$ is 1-3-6-7 and $l(P_1)=28$.

P_2 : 7, pred(7(2))=6(2), pred(6(2))=5(1), pred(5(1))=4(1), pred(4(1))=2(1), pred(2(1))=1(1).
$\Rightarrow P_2$ is 1-2-4-5-6-7 and $l(P_2)=28$.

P_3 : 7, pred(7(3))=4(1), pred(4 (1))=2(1), pred(2(1))=1(1).
$\Rightarrow P_3$ is 1-2-4-7 and $l(P_3)=27$

COMPLEXITY OF THE K-LONGEST PATH ALGORITHM

Consider m arcs and n nodes in the network. The complexity of the k-longest path algorithm is influenced by how it finds the maximum value and the number of repetitions (iterations) that occur in the main step of the algorithm.

In the main step, there are n-1 iterations, where each iteration corresponds to each node j= 2,3, ...,n respectively. Moreover, to determine the r-th longest path incident to a node j, at most r | B(j) | comparisons are to be made in order to find the maximum value for d_{jr}, where | B(j) | is the number of arcs incident into j. Thus, the total operation can be approximated by:

$$\sum_{j=2}^{n}\left(\sum_{r=1}^{k} r|B(j)|\right) = \left(\sum_{r=1}^{k} r\right)\left(\sum_{j=2}^{n}|B(j)|\right) = \frac{k(k+1)}{2}m$$

Hence the complexity of the k-longest path algorithm is $0(k^2m)$.

SOME USES OF K-LONGEST PATH

Here we consider some of the advantages of determining the k-longest paths to a project planning and scheduling problem. For a given integer k $\geq$ 2 and a project network (network model of a project), the ranked k-longest paths together with their respective lengths can be found using the k-longest path algorithm. In this case, the r-th components of the k-vector of distance label

$d(j) = (d_{j1}, d_{j2}, \ldots, d_{jk})$ that can be obtained by the algorithm gives us the least possible total duration for the completion of activities lying on the r-th longest path in the project network from the source node to a node j.

Theoretically, one of the advantages of the CPM is that it helps to focus attention on the critical path(s) where any slippage can delay completion of the project. But, having all critical paths, the next longest path may also have a potential to delay the completion of the project if not managed properly. Hence for more efficiency in managing a project, it is also helpful to know the set of ranked k-longest paths in a project network. By determining these k-longest paths that are suspected to have a potential to delay the completion of the project, the project manager can pay attention to them and make proper follow up of these paths so as to accomplish every activity on scheduled time as much as possible.

Knowledge of k-longest paths and their respective length (duration) helps also to answer questions like "how much is the duration of a path next to a critical path near to the optimal completion time of the project? How much difference is there between the duration of the second, third, ..., k-th longest path?" and "how much total float time is available for each of these k-longest paths?" This information can help for efficient utilisation of resources and also indicates to the project manager how much attention should be given to activities on respective paths.

Suppose it is required to complete a project in some targeted time T before the normal minimum completion time in order to meet a desired project due date. It is commonly recognised that the duration of the majority of activities in most projects can be reduced (called crashing) by allocating to it extra resources (manpower, capital, machine, etc.). To make crashing, reducing duration of some activities on the critical path(s) with minimum crashing cost constitutes the first action. However, the reduction of the duration of the critical path(s) only may not be sufficient since the next longest path can require a duration beyond T. Hence the knowledge of k-longest paths is helpful also in such cases. After finding k-longest paths $P_1, P_2, \ldots, P_k$ for which $l(P_{k-1}) > T \geq l(P_k)$, one can make economically the reduction of the duration of some activities that lie only on these ranked (k-1)-longest paths so as to meet the required project due date.

As a practical example, the precedence relationship among jobs in the project (ideal) "Building a hydroelectric power station" is given in Table 2.

This is followed by its project network representation and the application of the k-longest path algorithm to determine the 3-ranked longest paths in the project network.

Table 2. Precedence relationships among jobs and their durations for the hydroelectric power station building project. The number(s) under the column Immed. pred. indicate(s) the job number of immediate predecessor of the corresponding job.

Job No.	Job Description	Immed. Pred.	Arc repres.	Duration (weeks)
1	Ecological survey of dam site	—	(1,2)	10
2	File environmental impact report	—	(1,3)	7
3	Economic feasibility study	2	(3,4)	5
4	Get approval of a relevant authority	3	(4,5)	7
5	Preliminary design and cost estimation	1,4	(5,7)	6
6	Study alternative labour power	1,4	(5,6)	4
7	Project approval and commitment of funds	5,6	(7,8)	9
8	Call quotations for electrical equipment-(turbines, generators,...)	7	(8,10)	5
9	Select suppliers for electrical equipment	8	(10,11)	10
10	Final design of project	7	(8,9)	3
11	Select construction contractors	7	(8,12)	5
12	Arrange constriction material supply	10,11	(12,13)	2
13	Dam building	12	(13,14)	22
14	Power station building	12	(13,15)	16
15	Power lines erection	9,10	(11,17)	20
16	Electric equipment installation	9,14	(15,16)	8
17	Build up reservoir water level	13	(14,16)	4
18	Commission the generators	16,17	(16,17)	3
19	Start supply power	15, 18	(17,18)	2

The precedence relationships among the jobs of the hydroelectric power station building project can be represented as a directed network as follows (Fig. 2):

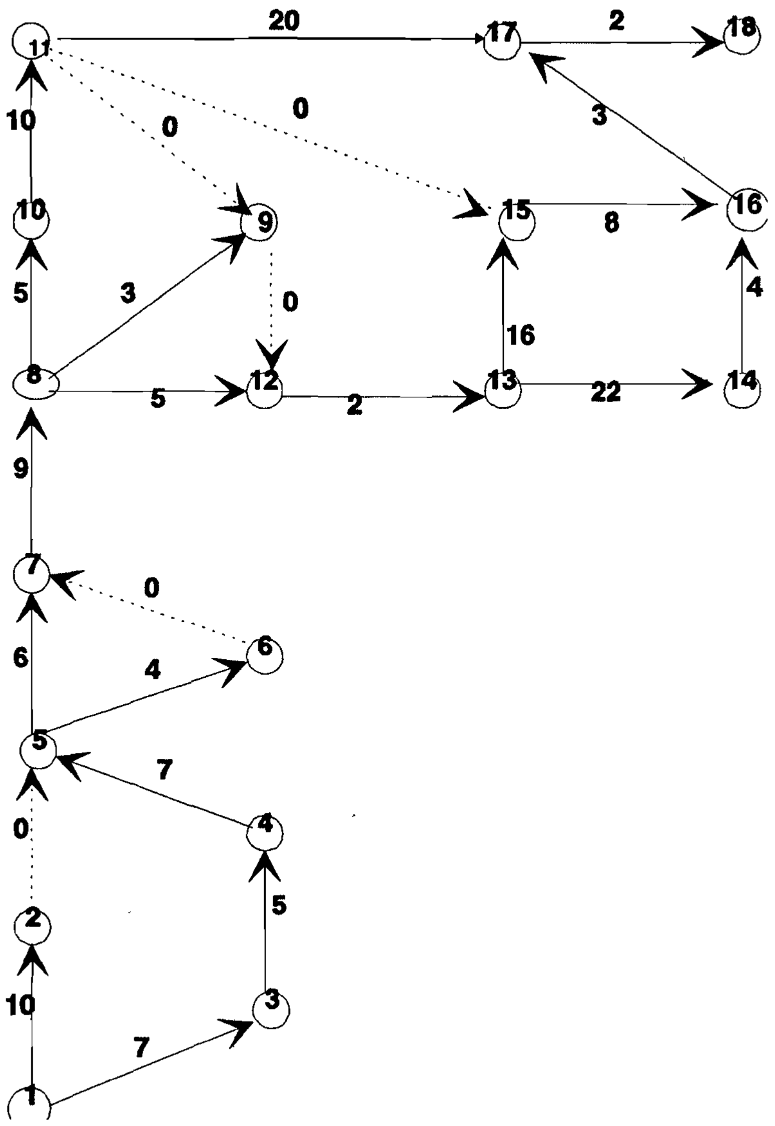

Fig. 2. Network diagram for the hydroelectric power station building problem.

Applying the ranked k-longest path algorithm to this project network (Fig. 2) for k = 3, we get:

First longest path: P_1: 1-3-4-5-7-8-12-13-14-16-17-18, and $l(P_1)=72$.

⇒ The minimum possible (optimal) duration of the project, under normal conditions, is 72 weeks and each activity on P_1 is a critical activity, i.e., if any activity (job) on this path is delayed by ϵ amount of time, then it results in a delay of completion time of the project by ϵ.

Second longest path: P_2:1-3-4-5-7-8-10-11-17-18; and $l(P_2)=71$.

Third longest path: P_3: 1-3-4-5-7-8-12-13-15-16-17-18; and $l(P_3)=70$.

⇒ There is a total float time of one and two weeks on the second and third ranked longest paths, respectively; i.e., completion of activities on the second and third ranked longest path can be delayed by up to one and two weeks respectively without affecting the optimal completion time of the project.

ACKNOWLEDGEMENT

I am very grateful to the German Academic Exchange Service (DAAD), which has been providing me with the relevant literature. I thank also anonymous reviewers of *SINET* for pointing out some mistakes in the original manuscript and for their valuable suggestions.

REFERENCES

1. Brucker, P. (1995). *Scheduling Algorithms*. Springer-Verlag, Berlin.
2. Hamacher, H.W. and Queyranne, M. (1985/6). K-best solutions to combinatorial optimisation problems. *Annals of Operations Research* 4:123–143.
3. Murty, K.G. (1992). *Network Programming*. Prentice-Hall Inc., New Jersey.
4. Turner, J.C. (1970). *Modern Applied Mathematics: Probability, Statistics, Operational Research*. Van Nostrand Reinhold Comp., New York.