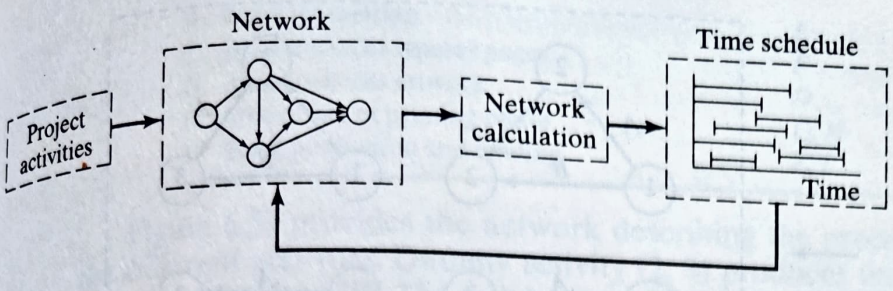


FIGURE 6.50
Phases for project planning with CPM-PERT

time requirements. Next, the project is translated into a network that shows the precedence relationships among the activities. The third step involves specific network computations that form the basis for the development of the time schedule for the project.

During the execution of the project, the schedule may not be realized as planned, causing some of the activities to be expedited or delayed. In this case, it will be necessary to update the schedule to reflect the realities on the ground. This is the reason for including a feedback loop between the time schedule phase and the network phase as shown in Figure 6.50.

The two techniques, CPM and PERT, which were developed independently, differ in that CPM assumes deterministic activity durations, whereas PERT assumes probabilistic durations. This presentation will start with CPM and then provide the details of PERT.

6.6.1 Network Representation

Each activity of the project is represented by an arc pointing in the direction of progress in the project. The nodes of the network establish the precedence relationships among the different activities of the project.

Two rules are available for constructing the network.

Rule 1. *Each activity is represented by one, and only one, arc.*

Rule 2. *Each activity must be identified by two distinct end nodes.*

Figure 6.51 shows how a dummy activity can be used to represent two concurrent activities, *A* and *B*. By definition, a dummy activity, which normally is depicted by a dashed arc, consumes no time or resources. Inserting a dummy activity in one of the four ways shown in Figure 6.51, we maintain the concurrence of *A* and *B*, and also provide unique end nodes for the two activities (to satisfy rule 2).

Rule 3. *To maintain the correct precedence relationships, the following questions must be answered as each activity is added to the network:*

(a) *What activities must immediately precede the current activity?*

(b) *What activities must follow the current activity?*

(c) *What activities must occur concurrently with the current activity?*

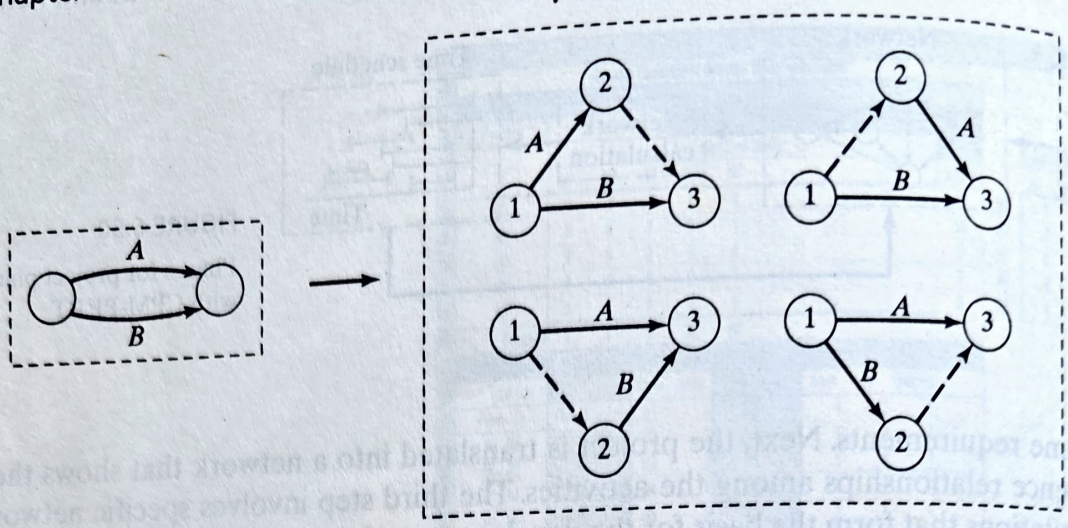


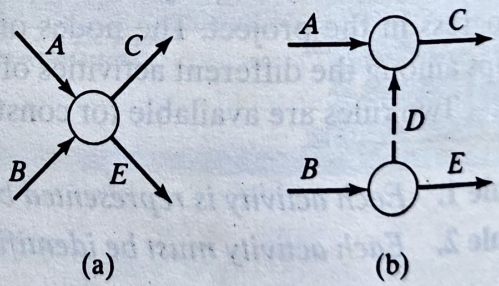
FIGURE 6.51
Use of dummy activity to produce unique representation of concurrent activities *A* and *B*

The answers to these questions may require the use of dummy activities to ensure correct precedences among the activities. For example, consider the following segment of a project:

1. Activity *C* starts immediately after *A* and *B* have been completed.
2. Activity *E* starts after *B* only has been completed.

Part (a) of Figure 6.52 shows the incorrect representation of the precedence relationship because it requires both *A* and *B* to be completed before *E* can start. In part (b), the use of a dummy activity rectifies the situation.

FIGURE 6.52
Use of dummy activity to ensure correct precedence relationship



Example 6.6-1

A publisher has a contract with an author to publish a textbook. The (simplified) activities associated with the production of the textbook are given below. Develop the associated network for the project.

Activity	Predecessor(s)	Duration (weeks)
A: Manuscript proofreading by editor	—	3
B: Sample pages prepared by typesetter	—	2
C: Book cover design	—	4
D: Preparation of artwork for book figures	—	3
E: Author's approval of edited manuscript and sample pages	*A*, *B*	2

Activity	Predecessor(s)	Duration (days)
A: Conduct feasibility study	—	3
B: Find potential buyer for present car	*A*	14
C: List possible models	*A*	1
D: Research all possible models	*C*	3
E: Conduct interview with mechanic	*C*	1
F: Collect dealer propaganda	*C*	2
G: Compile pertinent data	*D, E, F*	1
H: Choose top three models	*G*	1
I: Test-drive all three choices	*H*	3
J: Gather warranty and financing data	*H*	2
K: Choose one car	*I, J*	2
L: Choose dealer	*K*	2
M: Search for desired color and options	*L*	4
N: Test-drive chosen model once again	*L*	1
O: Purchase new car	*B, M, N*	3

6.6.2 Critical Path (CPM) Computations

The ultimate result in CPM is the construction of the time schedule for the project (see Figure 6.50). To achieve this objective conveniently, we carry out special computations that produce the following information:

1. Total duration needed to complete the project
2. Classification of the activities of the project as *critical* and *noncritical*

An activity is said to be **critical** if there is no "leeway" in determining its start and finish times. A **noncritical** activity allows some scheduling slack, so that the start time of the activity may be advanced or delayed within limits without affecting the completion date of the entire project.

To carry out the necessary computations, we define an **event** as a point in time at which activities are terminated and others are started. In terms of the network, an event corresponds to a node. Define

$$\square_j = \text{Earliest occurrence time of event } j$$

$$\Delta_j = \text{Latest occurrence time of event } j$$

$$D_{ij} = \text{Duration of activity } (i, j)$$

The definitions of the *earliest* and *latest* occurrence times of event j are specified relative to the start and completion dates of the entire project.

The critical path calculations involve two passes: The **forward pass** determines the *earliest* occurrence times of the events, and the **backward pass** calculates their latest occurrence times.

Forward Pass (Earliest Occurrence Times, $\square$). The computations start at node 1 and advance recursively to end node n.

Initial Step. Set $\square_1 = 0$ to indicate that the project starts at time 0.

General Step *j*. Given that nodes $p, q, \ldots$, and v are linked *directly* to node j by incoming activities $(p, j), (q, j), \ldots$, and (v, j) and that the earliest occurrence times of events (nodes) $p, q, \ldots$, and v have already been computed, then the earliest occurrence time of event j is computed as

$$\square_j = \max\{\square_p + D_{pj}, \square_q + D_{qj}, \ldots, \square_v + D_{vj}\}$$

The forward pass is complete when $\square_n$ at node n has been computed. By definition $\square_j$ represents the longest path (duration) to node j.

Backward Pass (Latest Occurrence Times, Δ). Following the completion of the forward pass, the backward pass computations start at node n and end at node 1.

Initial Step. Set $\Delta_n = \square_n$ to indicate that the earliest and latest occurrences of the last node of the project are the same.

General Step *j*. Given that nodes $p, q, \ldots$, and v are linked *directly* to node j by *outgoing* activities $(j, p), (j, q), \ldots$, and (j, v) and that the latest occurrence times of nodes $p, q, \ldots$, and v have already been computed, the latest occurrence time of node j is computed as

$$\Delta_j = \min\{\Delta_p - D_{jp}, \Delta_q - D_{jq}, \ldots, \Delta_v - D_{jv}\}$$

The backward pass is complete when Δ_1 at node 1 is computed.

Based on the preceding calculations, an activity (i, j) will be *critical* if it satisfies three conditions.

1. $\Delta_i = \square_i$
2. $\Delta_j = \square_j$
3. $\Delta_j - \Delta_i = \square_j - \square_i = D_{ij}$

The three conditions state that the earliest and latest occurrence times of nodes i and j are equal, and the duration D_{ij} fits "tightly" in the specified time span. An activity that does not satisfy all three conditions is *noncritical*.

The critical activities of a network must constitute an uninterrupted path that spans the entire network from start to finish.

Example 6.6-2

Determine the critical path for the project network in Figure 6.54. All the durations are in days.

Forward Pass

Node 1. Set $\square_1 = 0$

Node 2. $\square_2 = \square_1 + D_{12} = 0 + 5 = 5$

Node 3. $\square_3 = \max\{\square_1 + D_{13}, \square_2 + D_{23}\} = \max\{0 + 6, 5 + 3\} = 8$
Node 4. $\square_4 = \square_2 + D_{24} = 5 + 8 = 13$
Node 5. $\square_5 = \max\{\square_3 + D_{35}, \square_4 + D_{45}\} = \max\{8 + 2, 13 + 0\} = 13$
Node 6. $\square_6 = \max\{\square_3 + D_{36}, \square_4 + D_{46}, \square_5 + D_{56}\}$
$= \max\{8 + 11, 13 + 1, 13 + 12\} = 25$

The computations show that the project can be completed in 25 days.

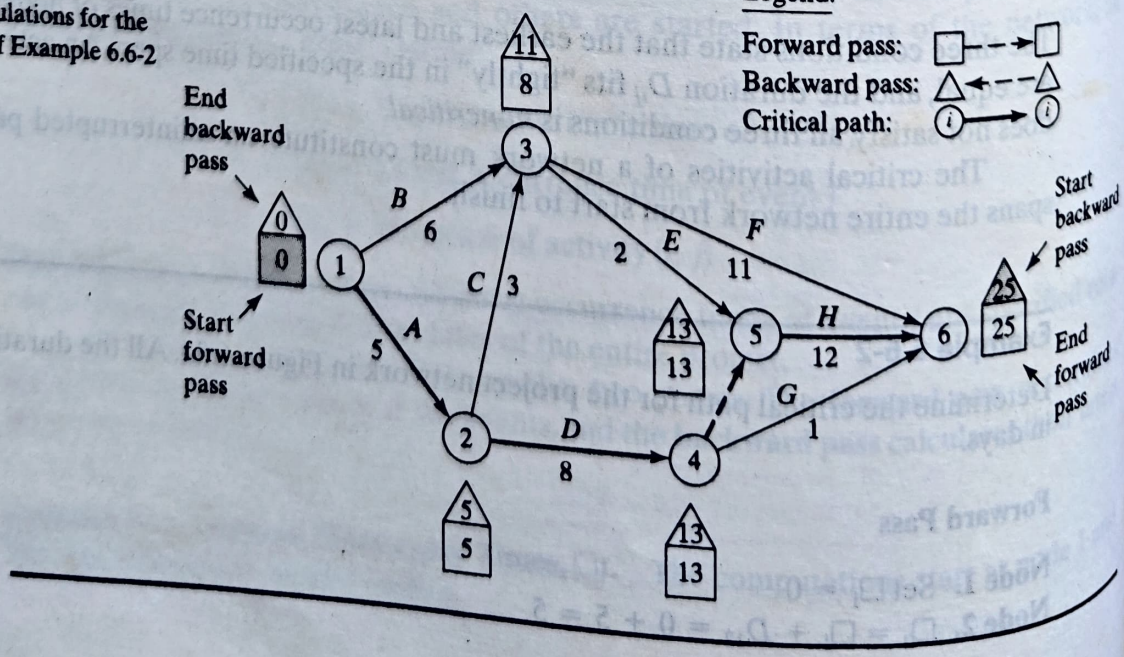
Backward Pass

Node 6. Set $\Delta_6 = \square_6 = 25$
Node 5. $\Delta_5 = \Delta_6 - D_{56} = 25 - 12 = 13$
Node 4. $\Delta_4 = \min\{\Delta_6 - D_{46}, \Delta_5 - D_{45}\} = \min\{25 - 1, 13 - 0\} = 13$
Node 3. $\Delta_3 = \min\{\Delta_6 - D_{36}, \Delta_5 - D_{35}\} = \min\{25 - 11, 13 - 2\} = 11$
Node 2. $\Delta_2 = \min\{\Delta_4 - D_{24}, \Delta_3 - D_{23}\} = \min\{13 - 8, 11 - 3\} = 5$
Node 1. $\Delta_1 = \min\{\Delta_3 - D_{13}, \Delta_2 - D_{12}\} = \min\{11 - 6, 5 - 5\} = 0$

Correct computations will always end with $\Delta_1 = 0$.

The forward and backward pass computations are summarized in Figure 6.54. The rules for determining the critical activities show that the critical path is defined by $1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$, which spans the network from start (node 1) to finish (node 6). The sum of the durations of the critical activities [(1, 2), (2, 4), (4, 5), and (5, 6)] equals the duration of the project (= 25 days). Observe that activity (4, 6) satisfies the first two conditions for a critical activity ($\Delta_4 = \square_4 = 13$ and $\Delta_5 = \square_5 = 25$) but not the third ($\square_6 - \square_4 \neq D_{46}$). Hence, the activity is not critical.

FIGURE 6.54
Forward and backward pass calculations for the project of Example 6.6-2



PROBLEM SET 6.6B

1. Determine the critical path for the project network in Figure 6.55.

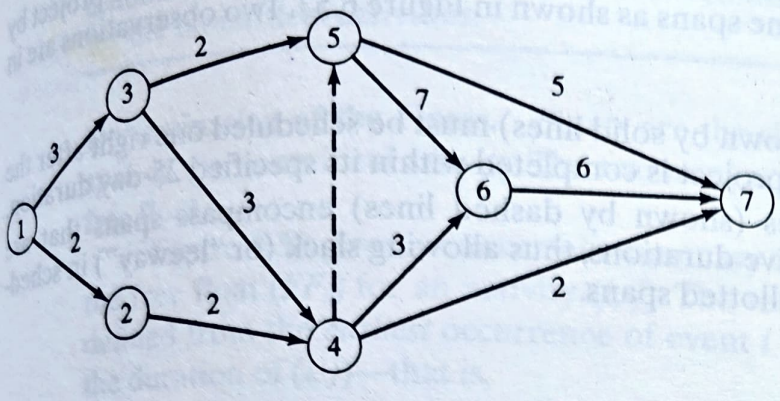
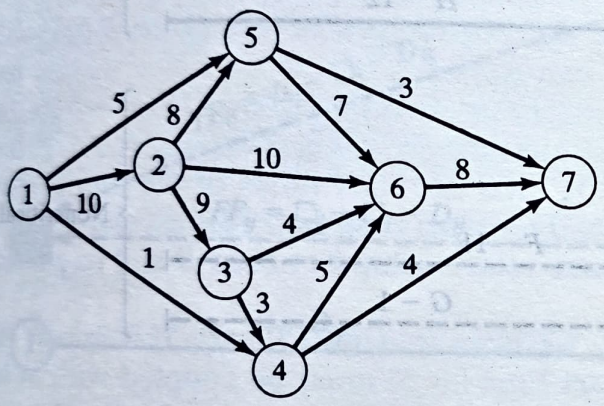
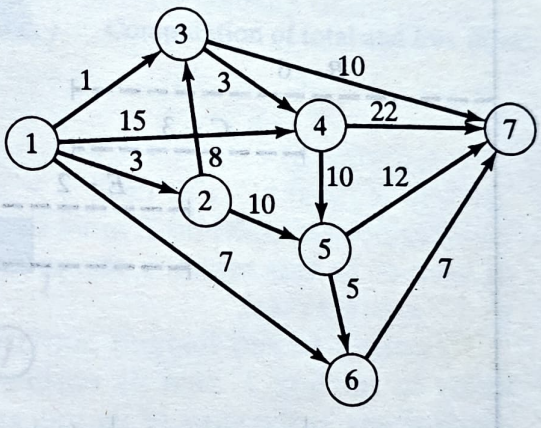


FIGURE 6.55
Project network for Problem 1, Set 6.6b

2. Determine the critical path for the project networks in Figure 6.56.



Project (a)



Project (b)

FIGURE 6.56
Project network for Problem 2, Set 6.6b

3. Determine the critical path for the project in Problem 6, Set 6.6a.
4. Determine the critical path for the project in Problem 8, Set 6.6a.
5. Determine the critical path for the project in Problem 9, Set 6.6a.
6. Determine the critical path for the project in Problem 10, Set 6.6a.

6.6.3 Construction of the Time Schedule

This section shows how the information obtained from the calculations in Section 6.6.2 can be used to develop the time schedule. We recognize that for an activity (i, j), $\square_i$ represents the *earliest start time,* and Δ_j represents the *latest completion time.* This means that $(\square_i, \Delta_j)$ delineates the (maximum) span during which activity (i, j) may be scheduled.

Construction of Preliminary Schedule. The method for constructing a preliminary schedule is illustrated by an example.

Example 6.6-3

Determine the time schedule for the project of Example 6.6-2 (Figure 6.54).

We can get a preliminary time schedule for the different activities of the project by delineating their respective time spans as shown in Figure 6.57. Two observations are in order.

1. The critical activities (shown by solid lines) must be scheduled one right after the other to ensure that the project is completed within its specified 25-day duration.
2. The noncritical activities (shown by dashed lines) encompass spans that are larger than their respective durations, thus allowing slack (or "leeway") in scheduling them within their allotted spans.

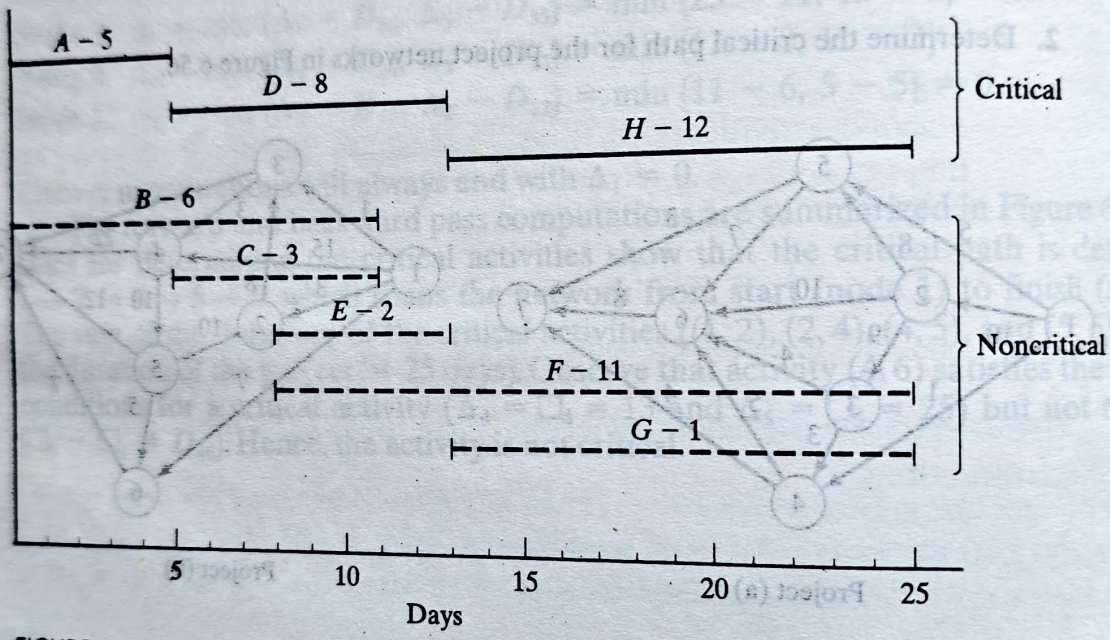


FIGURE 6.57

Preliminary schedule for the project of Example 6.6-2

How should we schedule the noncritical activities within their respective spans? Normally, it is preferable to start each noncritical activity as early as possible. In this manner, slack periods will remain opportunely available at the end of the allotted span, where they can be used to absorb unexpected delays in the execution of the activity. It may be necessary, however, to delay the start of a noncritical activity past its earliest time. For example, in Figure 6.57, suppose that each of the noncritical activities E and F requires the use of a bulldozer, and that only one is available. Scheduling both E and F as early as possible requires two bulldozers between times 8 and 10. We can remove the overlap by starting E at time 8 and pushing the start time of F to somewhere between times 10 and 14.

If all the noncritical activities can be scheduled as early as possible, the resulting schedule automatically is feasible. Otherwise, some precedence relationships may be violated if noncritical activities are delayed past their earliest time. Take, for example, activities C and E in Figure 6.57. In the project network (Figure 6.54), though C must

be completed before E, the spans of C and E in Figure 6.57 allow us to schedule C between times 6 and 9, and E between times 8 and 10. These spans, however, do not ensure that C will precede E. The need for a "red flag" that automatically reveals schedule conflict is thus evident. Such information is provided by computing the *floats* for the noncritical activities.

Determination of the Floats. Floats are the slack times available within the allotted span of the noncritical activity. The two most common floats are the **total float** and the **free float**.

Figure 6.58 gives a convenient summary for computing the total float (TF_{ij}) and the free float (FF_{ij}) for an activity (i, j). The total float is the excess of the time span defined from the *earliest* occurrence of event i to the *latest* occurrence of event j over the duration of (i, j)—that is,

$$TF_{ij} = \Delta_j - \square_i - D_{ij}$$

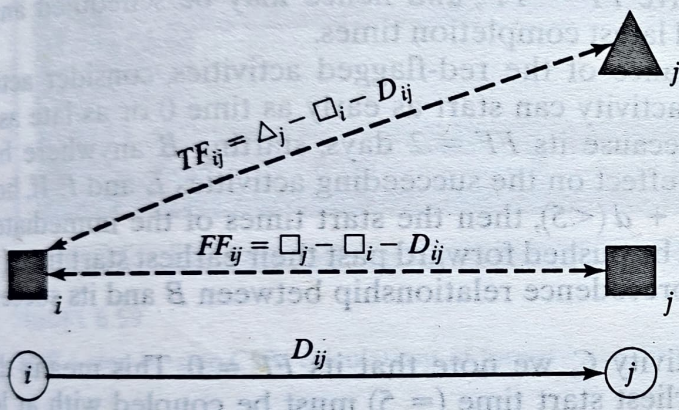


FIGURE 6.58
Computation of total and free floats

The free float is the excess of the time span defined from the *earliest* occurrence of event i to the *earliest* occurrence of event j over the duration of (i, j)—that is,

$$FF_{ij} = \square_j - \square_i - D_{ij}$$

By definition, $FF_{ij} \leq TF_{ij}$.

Red-Flagging Rule. *For a noncritical activity (i, j)*

(a) *If $FF_{ij} = TF_{ij}$, then the activity can be scheduled anywhere within its $(\square_i, \Delta_j)$ span without causing schedule conflict.*

(b) *If $FF_{ij} < TF_{ij}$, then the start of activity (i, j) can be delayed by at most FF_{ij} relative to its earliest start time $(\square_i)$ without causing schedule conflict. Any delay larger than FF_{ij} (but not more than TF_{ij}) must be accompanied by an equal delay relative to $\square_j$ in the start time of all the activities leaving node j.*

The implication of the rule is that a noncritical activity (i, j) will be red-flagged if its $FF_{ij} < TF_{ij}$. This red flag is important only if we decide to delay the start of the activity past its earliest start time, $\square_i$, in which case we must pay attention to the start times of the activities leaving node j to avoid schedule conflicts.

14.7 ROLE OF NETWORK TECHNIQUES IN PROJECT MANAGEMENT

The complexities of the present-day management problems and the business competitions have added to the pressure on the brains of decision-makers. In a large and complex project involving a number of interrelated activities, requiring a number of men, machines and materials, it is not possible for the management to make and execute an optimum schedule just by intuition based on the organisational capabilities and work experience. Managements are, thus, always on the look out for some methods and techniques which may help in planning, scheduling and controlling the project. The aim of planning is to develop a sequence of activities of the project, so that the project completion time and cost are properly balanced and the excessive demand of key resources is avoided. To meet the object of systematic planning, the managements have evolved a number of techniques applying network strategy. As already explained, PERT and CPM are two such widely applied techniques used for planning, scheduling and controlling of large and complex projects. With slight modifications both have given rise to several other network techniques, such as PEP (Programme Evaluation Procedure), RAMPS (Resource Allocation for Multi-Project Scheduling), LESS (Least Cost Estimating and Scheduling) and SCANS (Scheduling and Control by Automated Network System), etc.

14.8 NETWORK LOGIC (NETWORK OR ARROW DIAGRAM)

Some of the terms commonly used in networks are defined below.

Activity

It is physically identifiable part of a project which requires time and resources for its execution. An activity is represented by an arrow, the tail of which represents the start and the head, the finish of the activity. The length, shape and direction of the arrow has no relation to the size of the activity.

Event

The beginning and end points of an activity are called events or nodes. Event is a point in time and does not consume any resources. It is represented by a circle. The head event, called the jth event, has always a number higher than the tail event, called the ith event *i.e.*, $j > i$. For example

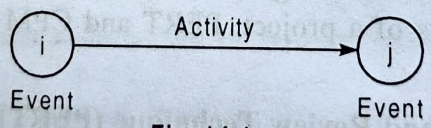


Fig. 14.1

'Making the pattern of impeller' is an activity.
'Start making the pattern of impeller' is an event.
'Pattern making completed' is an event.

Path

An unbroken chain of activity arrows connecting the initial event to some other event is called a path.

Network

It is the graphical representation of logically and sequentially connected arrows and nodes representing activities and events of a project. Networks are also called *arrow diagrams*.

Network Construction

Firstly the project is split into activities. Start and finish events of the project are then decided. After deciding the precedence order, the activities are put in a logical sequence by using the graphical notations. While constructing the network, in order to ensure that the activities fall in a logical sequence, following questions are checked:

(*i*) What activities must be completed before a particular activity starts ?
(*ii*) What activities follow this ?
(*iii*) What activities must be performed concurrently with this ?

Activities which must be completed before a particular activity starts are called the *predecessor activities* and those which must follow a particular activity are called *successor activities*.

While drawing the network following points should be kept in mind:

1. Each activity is represented by one and only one arrow. But in some situations where an activity is further subdivided into segments, each segment will be represented by a separate arrow.
2. Time flows from left to right. Arrows pointing in opposite direction are to be avoided.
3. Arrows should be kept straight and not curved.
4. Angles between the arrows should be as large as possible.
5. Arrows should not cross each other. Where crossing cannot be avoided, bridging should be done as shown in Fig. 14.6.
6. Each activity must have a tail and a head event. No two or more activities may have the same tail and head events.
7. An event is not complete until all the activities flowing into it are completed.
8. No subsequent activity can begin until its tail event is completed.
9. In a network diagram there should be only one initial event and one end event.

Dummy

An activity which only determines the dependency of one activity on the other, but does not consume any time is called a dummy activity. Dummies are usually represented by dotted line arrows.

To illustrate the use of dummy, refer to Fig. 14.2 (*a*) and assume that the start of activity C depends upon the completion of activities A and B and that the start of activity E depends only on the completion of activity B. For this situation, figure 14.2 (*a*) is a faulty representation. This is corrected by introducing a dummy activity D as shown in Fig. 14.2 (*b*).

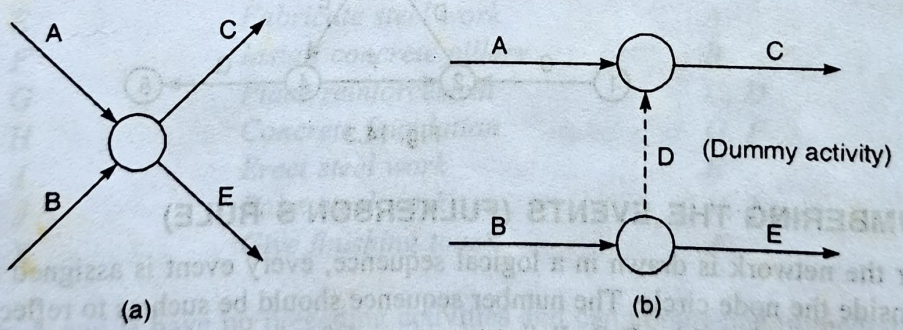


Fig. 14.2. (a), (b)

A dummy activity is introduced in the network for two basic reasons:

1. To maintain the precise logic of the precedence of activities. Such a dummy is called '*logical dummy*'. It is shown in Fig. 14.2 (*b*).
2. To comply with the rule that no two or more activities can have the same tail and head events. Such a dummy is called '*grammatical dummy*'. In Fig. 14.2 (*c*), both activities A and B have the same tail event 10 and same head event 20, which is incorrect since no two activities can have the same pair of tail and head events. Such activities are called duplicate activities. This difficulty is resolved by the introduction of a dummy activity in any of the four ways represented in Fig. 14.2 (*d*), (*e*), (*f*) or (*g*).

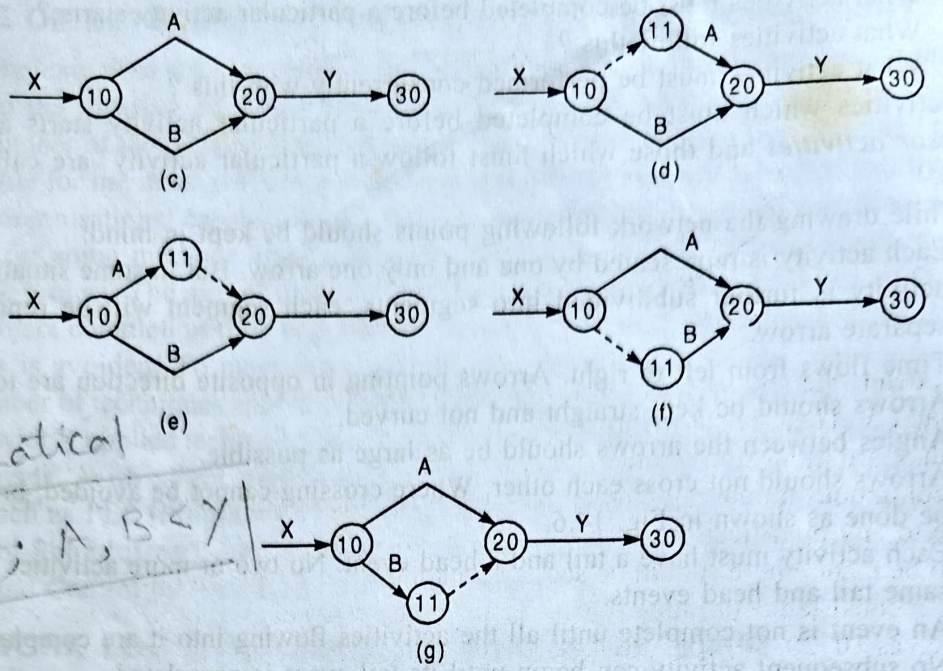


Fig. 14.2. (c) to (g)

Looping (Cycling)

Sometimes due to faulty network sequence a condition illustrated in figure 14.3, arises. Here the activities D, E and F form a loop (cycle). Activity D cannot start until F is completed, which, in turn, depends upon the completion of E. But E is dependent upon the completion of D. Thus the network cannot proceed. This situation can be avoided by checking the precedence relationship of the activities and by numbering them in a logical sequence.

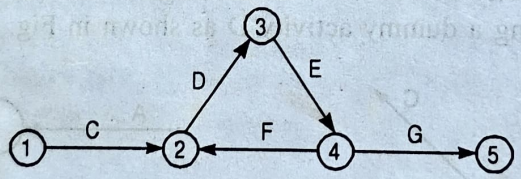


Fig. 14.3

14.9 NUMBERING THE EVENTS (FULKERSON'S RULE)

After the network is drawn in a logical sequence, every event is assigned a number which is placed inside the node circle. The number sequence should be such as to reflect the flow of the network. The rule devised by D.R. Fulkerson is used for the purpose of numbering. It involves the following steps;

1. The initial event which has all outgoing arrows with no incoming arrow is numbered '1'.
2. Delete all the arrows coming out from node '1'. This will convert some more nodes (at least one) into initial events. Number these events 2, 3,...
3. Delete all the arrows going out from these numbered events to create more initial events. Assign the next numbers to these events.
4. Continue until the final or terminal node, which has all arrows coming in with no arrow going out, is numbered.

To illustrate the numbering technique let us consider the network shown in figure. 14.4. Event A is initial event and is numbered 1. Delete the arrows *a* and *b*. This will create two more (B & C) initial events. Number these 2 & 3. Now delete the arrows coming out from nodes

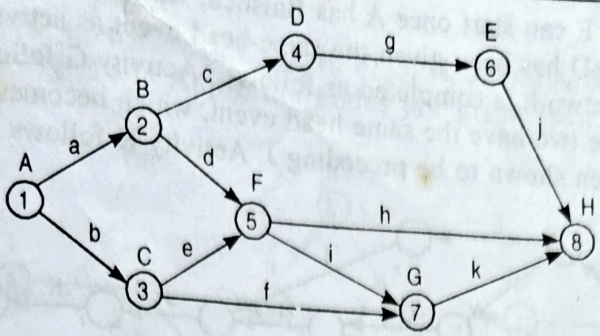


Fig. 14.4

2 and 3, *i.e.*, delete the arrows *c*, *d*, *e* and *f*. This converts D and F into initial events. Number these nodes as 4 and 5. Then delete the arrows g, *h* and *i*, the two nodes E and G become the initial nodes and they are numbered 6 and 7. Then the last event or terminal event is numbered as 8.

This continuous numbering may be all right when the project is very small and the network is not liable to any modifications later on. But in large networks, where extensive modification may have to be made, there should be scope of adding more events and numbering them without causing any inconsistency or loops. This is achieved by *skip numbering*. One way is to assign the numbers such as 10, 20, 30, 40,..., or 4, 8, 12,16,..., etc. The second way is to leave some numbers such as 7, 8, 9; 17, 18, 19; 27, 28, 29;... and allot them to the events added afterwards. There can be still more ways of doing skip numbering.

EXAMPLE 14.1

Draw a network for the simple project of erection of steel works for a shed. The various activities of the project are as under:

Activity	*Description*	*Preceded by*
A	*Erect site workshop*	—
B	*Fence site*	—
C	*Bend reinforcement*	*A*
D	*Dig foundation*	*B*
E	*Fabricate steel work*	*A*
F	*Install concrete pillars*	*B*
G	*Place reinforcement*	*C, D*
H	*Concrete foundation*	*G, F*
I	*Erect steel work*	*E*
J	*Paint steel work*	*H, I*
K	*Give finishing touch*	*J*

Solution

(*i*) Activities A and B have no preceding activities and can commence immediately [Fig. 14.5 (*a*)].

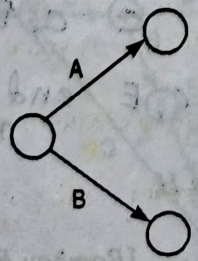


Fig. 14.5 (a)

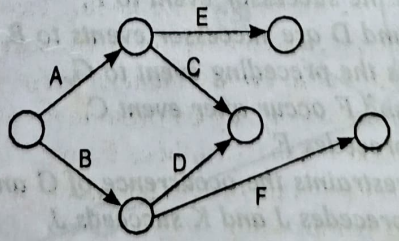


Fig. 14.5 (b)

(*ii*) Activities C and E can start once A has finished, while D and F can start once B has finished. Note that activity D has been given the same head event as activity C in Fig. 14.5 (*b*).

(*iii*) The remaining network is completed as follows : Activity G follows C and D. Since H is preceded by G and F, the two have the same head event, which becomes the tail event for H. Similarly, I and H have been shown to be preceding J. Activity K follows J [Fig. 14.5 (*c*)].

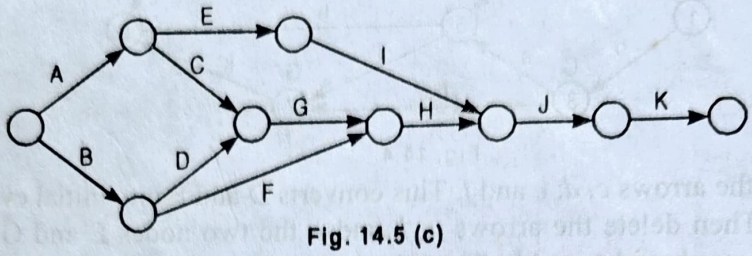


Fig. 14.5 (c)

EXAMPLE 14.2

A project consists of a series of tasks labelled A, B, ..., H, I with the following relationships (W < X, Y means X and Y cannot start until W is completed; X, Y < W means W cannot start until both X and Y are completed). With this notation construct the network diagram having the following constraints:

$$A < D, E;\ B, D < F;\ C < G;\ B < H;\ F, G < I.$$

[P.U.B.Com. Sept., 2005]

Solution

For the given precedence relationships, the project network shown in Fig. 14.6 is obtained. Since H is preceded by B while F is preceded by B and D, a dummy activity must be incorporated to draw the network.

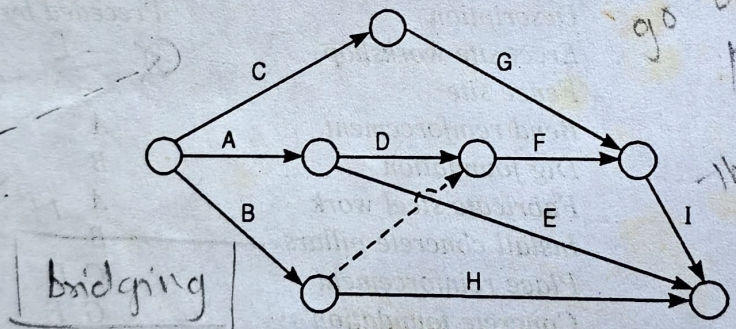


Fig. 14.6

EXAMPLE 14.3

Draw a network for the following project and number the events according to Fulkerson's rule:

A is the start event and K is the end event,
A precedes event B,
J is the successor event to F,
C and D are successor events to B,
D is the preceding event to G,
E and F occur after event C,
E precedes F,
C restraints the occurrence of G and G precedes H,
H precedes J and K succeeds J,
F restraints the occurrence of H.

[Bombay B.Sc. (Stat.) 1974]

EXAMPLE 14.5

Depict the following dependency relationships by means of network diagrams. The alphabets stand for activities.

(i) A and B control F; B and C control G.

(ii) A and B control F; B controls G while C controls G and H.

(iii) A controls F and G; B controls G while C controls G and H.

(iv) F and G are controlled by A; G and H are controlled by B with H controlled by B and C.

(v) A controls F, G and H; B controls G and H with H controlled by C.

Solution

The dependency relationships are depicted by Fig. 14.9 (i) through (v). The use of dummy activities is to be noted.

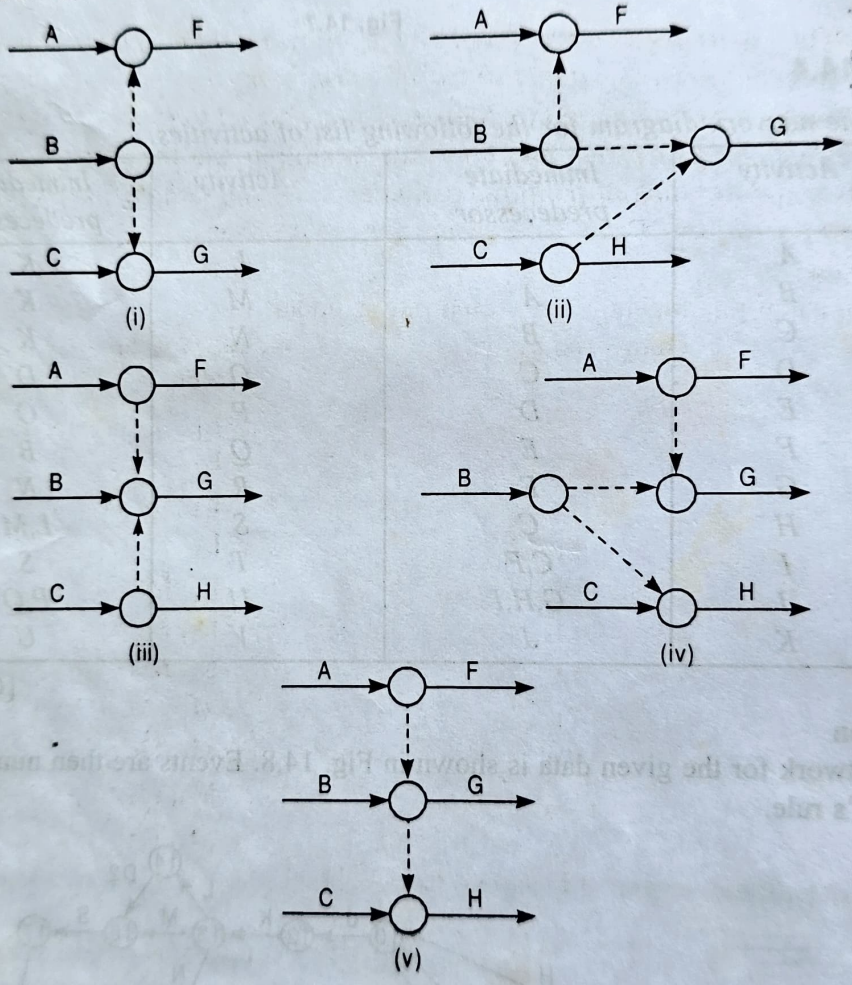


Fig. 14.9 (i) to (v)

EXAMPLE 14.6

Construct a network diagram for a project comprising of activities B, C, E, F, G, H, I, J, L, M, N, P and Q such that the following precedence relationships are satisfied:

B < E, F; C, F < G; C < L; E, G < H; H, L < I; H < J; L < M; H, M < N; I, J < P; N < Q.

[J.U. B.E.(Mech.) 2004; Andhra M.Sc. (Stat.) 1979, Osmania M.Sc. (Math.) 1984]

Solution

The network is shown in Fig. 14.10. Four dummy activities have been used. Since activity L depends upon C. and G depends upon both F and C, dummy D_1 is required. Dummies D_2 and

This is the main difference between the two techniques. The other difference between the two is that PERT is event-oriented while the CPM is activity-oriented. In Fig. 14.20 (*a*) and (*b*),

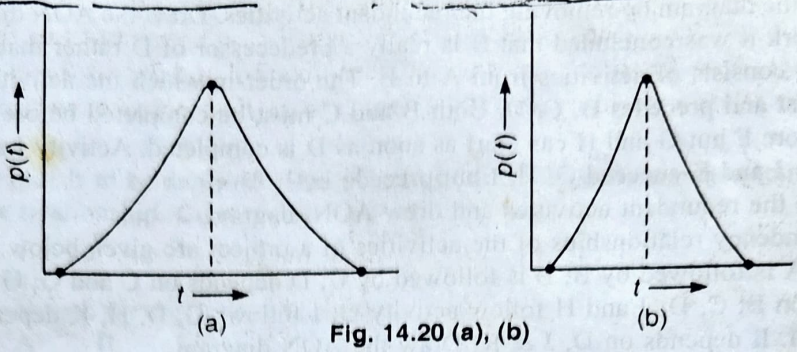


Fig. 14.20 (a), (b)

t = Activity time,
$p(t)$ = relative frequency of occurrence.

Time Units

Any convenient time unit can be used, but it must be consistent throughout the network. Depending upon the project length and level of detail, time unit may be working days, shifts or weeks. Full time units are usually used, for instance activity estimated at 3 days and 6 hours will be assigned 4 days.

Critical Path Analysis

The Critical path of a network gives the shortest time in which the whole project can be completed. It is the chain of activities with the longest time durations. These activities are called *critical activities*. They are critical in the sense that delay in any of them results in the delay of the completion of the project. There may be more than one critical path in a network and it is possible for the critical path to run through a dummy. The critical path analysis consists of the following steps:

1. *Calculate the time schedule for each activity* : It involves the determination of the time by which an activity must begin and the time before which it must be completed. The time schedule data for each activity include the calculation of the earliest start, the earliest finish, the latest start, the latest finish times and the float.
2. *Calculate the time schedule for the completion of the entire project* : It involves the calculation of project completion time.
3. *Identify the critical activities and find the critical path* : Critical activities are the ones which must be started and completed on schedule or else the project may get delayed. The path containing these activities is the critical path and is the longest path in terms of duration.

Consider the network shown in Fig. 14.21 which consists of the following activities:

Activity	:	1-2	1-3	2-3	2-5	3-4	3-6	4-5	4-6	5-6	6-7
Duration (weeks)	:	15	15	3	5	8	12	1	14	3	14

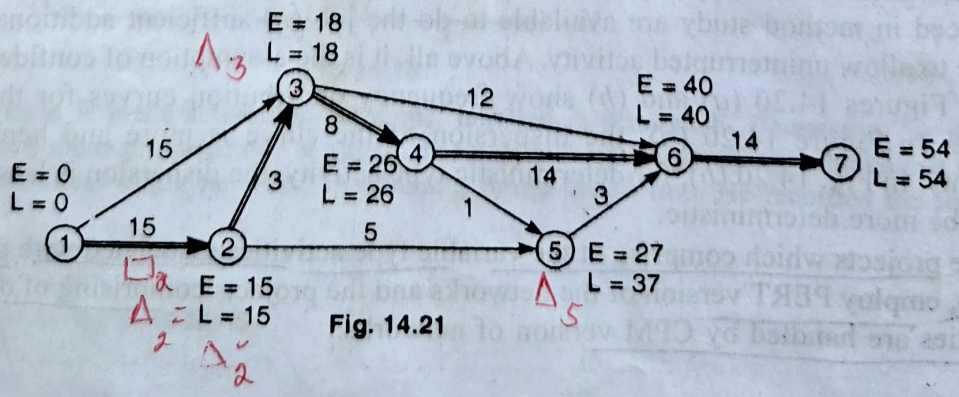


Fig. 14.21

The *earliest start time* (E) for an activity represents the time at which an activity can begin at the earliest. It assumes that all the preceding activities start and finish at their earliest times. For instance earliest start times of activities 1-2 and 1-3 are zero each or the earliest occurrence time of event 1 is zero. Earliest start times of activities 2-3 and 2-5 or the earliest occurrence time of event 2 is obtained by adding activity time t_{12} to earliest occurrence time of event 1 *i.e.*, it is $0 + 15 = 15$.

Next consider event 3. It can be reached directly from event 1 or via event 2, the times for the two sequences being 15 and $15 + 3 = 18$. Since event 3 can occur only when all the preceding activities and events have taken place, its earliest occurrence time or the earliest start times of activities emanating from even 3 is 18, the higher of the two values 15 and 18. This is represented by putting E = 18 around its node in the network. Likewise, the earliest occurrence time of each event can be determined by proceeding progressively from left to right *i.e.*, following the *forward pass method* according to the following rule:

If only one activity converges on an event, its earliest start time E is given by E of the tail event of the activity plus activity duration. If more than one activity converges on it, E's via all the paths would be computed and the highest value chosen and put around the node.

The E's calculated for the problem at hand are shown in the network diagram.

The *latest finish time* (L) for an activity represents the latest by which an activity must be completed in order that the project may not be delayed beyond its targeted completion time. This is calculated by proceeding progressively from the end event to the start event. The L for the last event is assumed to be equal to its E and the L's for the other events are computed by the following rule (using *backword pass method*):

If only one activity emanates from an event, compute L by subtracting activity duration from L of its head event. If more than one activity emanates from an event, compute L's via all the paths and choose the smallest and put it around the event at hand.

The L's calculated for the problem at hand are shown in the network diagram.

Next, the *earliest finish time* (T_{EF}) and the *latest start time* (T_{LS}) for an activity are computed:

$$T_{EF} = E + t_{ij},$$
$$T_{LS} = L - t_{ij},$$

where t_{ij} is the duration for activity i - j. Float (also called *total float*) for an activity is then calculated:

$$F = L - T_{EF} \quad \text{or} \quad F = T_{LS} - E.$$

Float is, thus, the positive difference between the finish times or the positive difference between the start times. The following analysis table is then compiled:

TABLE 14.1

Activity (i - j)	Duration (D)	Start time		Finish time		Total Float
		Earliest	Latest	Earliest	Latest	
(1)	(2)	(3)	(4)	(5)	(6)	(7)
1-2	15	0	0	15	15	0
1-3	15	0	3	15	18	3
2-3	3	15	15	18	18	0
2-5	5	15	32	20	37	17
3-4	8	18	18	26	26	0
3-6	12	18	28	30	40	10
4-5	1	26	36	27	37	10
4-6	14	26	26	40	40	0
5-6	3	27	37	30	40	10
6-7	14	40	40	54	54	0

Columns 1 and 2 contain the activities and their durations is weeks. Under column 3 are noted the E's for the tail events and under column 6 are noted the L's of the head events. Other columns are then computed as follows:

column 4 = column 6 – column 2,
column 5 = column 3 + column 2,
and column 7 = column 6 – column 5,
= column 4 – column 3.

As an example, consider activity 1–2. Its tail event 1 has E = 0. Put 0 against this activity under column 3. Its head event 2 has L = 15. Put 15 against it under column 6. Under column 4 note the value in column 6 minus activity duration *i.e.*, 15 – 15 = 0. Under column 5 note the value in column 3 plus activity duration *i.e.*, 0 + 15 = 15. Compute total float by subtracting column 5 from 6 or column 3 from 4. Total float for activity 1-2 is 0. Similarly, calculate total float for other activities. Critical path is the path containing activities with zero float. These activities demand above normal attention with no freedom of action. For the problem at hand it is 1-2-3-4-6-7 and is shown by double arrows in Fig. 14.21. The project duration is 54 weeks. Sometimes, there may be more than one critical path *i.e.*, two or more paths with the same maximum completion time. Non-critical activities have positive float (slack or leeway) so that we may slacken while executing them and concentrate on the critical activities. While delay in any critical activity will delay the project completion, this may not be so with the non-critical activities.

The Three Floats

Total float : It is the difference between the maximum time available to perform the activity and the activity duration. The maximum time available for any activity is from the earliest start time to the latest completion time. Thus for an activity i - j having duration t_{ij},

Maximum time available = L – E.

$$\therefore \quad \text{Total Float} = L - E - t_{ij}$$
$$= (L - t_{ij}) - E \text{ or } L - (E + t_{ij})$$
$$= T_{LS} - E \quad \text{or} \quad L - T_{EF}.$$

Thus the total float of an activity is the difference of its latest start and earliest start times or the difference of its latest finish and earliest finish times. Total float represents the maximum time within which an activity can be delayed without affecting the project completion time.

Free Float : It is that portion of the total float within which an activity can be manipulated without affecting the floats of subsequent activities. It is computed by subtracting the head event slack from the total float. The head event slack is (L – E) of the event.

∴ Free float of activity

i - j = T.F. – (L – E) of event j.

Thus free float is the time by which completion of an activity can be delayed without delaying its immediate successor activities.

Independent Float : It is that portion of the total float within which an activity can be delayed for start without affecting the floats of preceding activities. It is computed by subtracting the tail event slack from the free float. If the result is negative, it is taken as zero.

∴ Independent float of activity

i - j = F.F. – (L – E) of tail event i.

Apart from the above three floats, there is another float, namely the interfering float for the activities.

Interfering Float : Utilization of the float of an activity can affect the floats of the subsequent activities in the network. Thus, interfering float can be defined as that part of the total float which causes a reduction in the floats of the succeeding activities. In other words *it can be defined as the difference between the latest finish time of the activity under consideration and the earliest start time of the following activity, or zero, whichever is larger*. Thus, interfering float refers to that portion of the activity float which cannot be consumed without adversely affecting the floats of the subsequent activities.

It is numerically equal to the difference between the total float and the free float of the activity. It is also equal to the head event slack of the activity.

Thus interfering float of an activity = T.F. – F.F. = (L – E) of the head event of the activity.

Subcritical Activity : Activity having next higher float than the critical activity is called the subcritical activity and demands normal attention but allows some freedom of action. The path connecting such activities is named as the *subcritical path*. A network may have more than one subcritical path.

Supercritical Activity : An activity having negative float is called supercritical activity. Such an activity demands very special attention and action. It results when activity duration is more than the time available. Such negative float, though possible, indicates an abnormal situation requiring a decision as to how to compress the activity. It can be done by employing more resources so as to make the total float zero or positive. Compression of the network, however, involves an extra cost.

Slack : It is the time by which occurrence of an event can be delayed. It is denoted by S and is the difference between the latest occurrence time and earliest occurrence time of the event.

i.e., S = L – E of the event.

In the above discussion, the term float has been used in connection with the activities and slack for the events. However, the two terms are being used interchangeably *i.e.*, slack for the activities and float for the events by some of the writers.

EXAMPLE 14.12-1

Tasks A, B, C,..., H, I constitute a project. The precedence relationships are

$$A < D;\ A < E;\ B < F;\ D < F;\ C < G;\ C < H;\ F < I;\ G < I.$$

Draw a network to represent the project and find the minimum time of completion of the project when time, in days, of each task is as follows:

Task	:	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
Time	:	*8*	*10*	*8*	*10*	*16*	*17*	*18*	*14*	*9*

Also identify the critical path.

[*P.U. B.Com. Sept. 2005; Roorkee M.Sc. (App. Math.), 1976; Madurai M.Sc. (Appl. Sc.) 1983*]

Solution

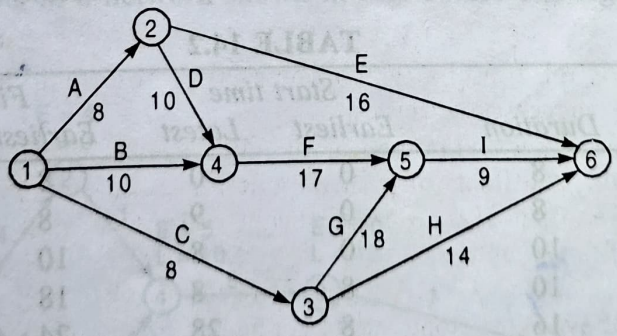


Fig. 14.22 (a)

The given precedence order reveals that there are no predecessors to activities A, B, and C and hence they all start from the initial node. Similarly, there are no successor activities to activities E, H and I and hence, they all merge into the end node of the project. The network obtained is shown in Fig. 14.22 (*a*).

The nodes of the network have been numbered by using the Fulkerson's rule. The activity descriptions and times are written along the activity arrows. To determine the minimum project

completion time, let event 1 occur at zero time. The earliest occurrence time (E) and the latest occurrence time (L) of each event is then computed.

$$E_1 = 0,$$
$$E_2 = E_1 + t_{12} = 0 + 8 = 8,$$
$$E_3 = E_1 + t_{13} = 0 + 8 = 8,$$
$$E_4 = \text{Max. } [0 + 10, 8 + 10] = 18,$$
$$E_5 = \text{Max. } [18 + 17, 8 + 18] = 35,$$
$$E_6 = \text{Max. } [8 + 16, 35 + 9, 8 + 14] = 44.$$

Similarly,

$$L_6 = E_6 = 44,$$
$$L_5 = L_6 - t_{56} = 44 - 9 = 35,$$
$$L_4 = L_5 - t_{45} = 35 - 17 = 18,$$
$$L_3 = \text{Min. } [44 - 14, 35 - 18] = 17,$$
$$L_2 = \text{Min. } [44 - 16, 18 - 10] = 8,$$
$$L_1 = \text{Min. } [8 - 8, 17 - 8, 18 - 10] = 0.$$

The E and L values for each event have been written along the nodes in Fig. 14.22 (*b*).

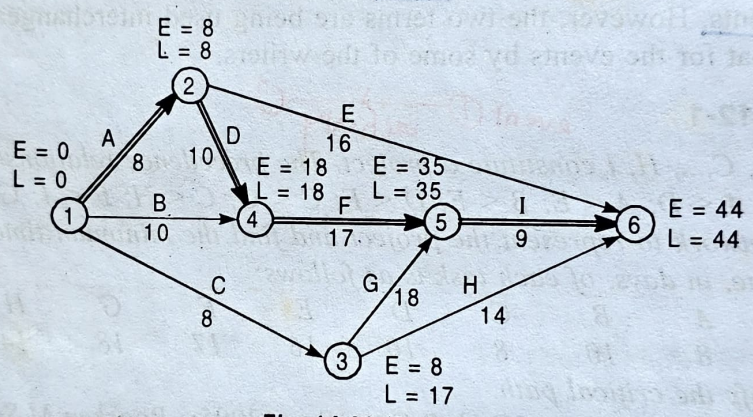


Fig. 14.22 (b)

The critical path is now determined by any of the following methods:

Method 1. The network analysis table is compiled as below:

TABLE 14.2

Activity	*Duration*	*Start time*		*Finish time*		*Total float*
		Earliest	*Latest*	*Earliest*	*Latest*	
1-2	8	0	0	8	8	0
1-3	8	0	9	8	17	9
1-4	10	0	8	10	18	8
2-4	10	8	8	18	18	0
2-6	16	8	28	24	44	20
3-5	18	8	17	26	35	9
3-6	14	8	30	22	44	22
4-5	17	18	18	35	35	0
5-6	9	35	35	44	44	0

Activities 1-2, 2-4, 4-5 and 5-6 having zero float are the critical activities and 1-2-4-5-6 is the critical path.

Method 2. For identifying the critical path, the following conditions are checked. If an activity satisfies all the three conditions, it is critical.

EXAMPLE 14.12-3

The utility data for a network are given below. Determine the total, free, independent and interfering floats and identify the critical path.

Activity :	*0-1*	*1-2*	*1-3*	*2-4*	*2-5*	*3-4*	*3-6*	*4-7*	*5-7*	*6-7*
Duration :	*2*	*8*	*10*	*6*	*3*	*3*	*7*	*5*	*2*	*8*

[H.P.U.B. Tech. (Mech.) Nov., 2006; P.U.B. Com. Sept., 2004; K.U.M. Tech. (Mech.) May, 1994]

Solution

The network diagram for the given project data is shown in Fig. 14.24. Activity durations are written along the activity arrows.

The earliest start and latest finish times of the activities are computed by employing the forward pass and backward pass calculations as explained in example 14.12-2. These times are represented in the network around the respective nodes.

The network analysis table is now constructed.

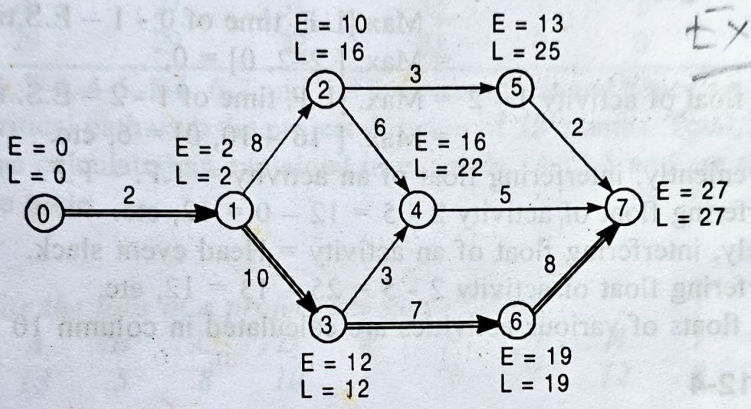


Fig. 14.24

TABLE 14.4

Activity	*Duration*	*Start time*		*Finish time*		*Float*			
		Earliest	*Latest*	*Earliest*	*Latest*	*Total*	*Free*	*Independent*	*Interfering*
1	2	3	4	5	6	7	8	9	10
0 - 1	2	0	0	2	2	0	0	0	0
1 - 2	8	2	8	10	16	6	0	0	6
1 - 3	10	2	2	12	12	0	0	0	0
2 - 4	6	10	16	16	22	6	0	$-6 \approx 0$	6
2 - 5	3	10	22	13	25	12	0	$-6 \approx 0$	12
3 - 4	3	12	19	15	22	7	1	1	6
3 - 6	7	12	12	19	19	0	0	0	0
4 - 7	5	16	22	21	27	6	6	0	0
5 - 7	2	13	25	15	27	12	12	0	0
6 - 7	8	19	19	27	27	0	0	0	0

Total float is the positive difference between latest and earliest finish times or latest and earliest start times. For activity 1-2,

total float (T.F.) = 16 – 10 = 8 – 2 = 6.

Similarly, for activity, say 2-5,

total float = 25 – 13 = 22 – 10 = 12 and so on.

Total float calculations are depicted in column 7 of table 14.4.

Free float of activity i - j = T.F. – head event slack

= T.F. – (L – E) of event j.

Thus free float of activity 0 - 1
= 0 – (L – E) of event 1,
= 0 – (2 – 2) = 0,
free float of activity 1 - 2 = 6 – (16 – 10) = 6 – 6 = 0, etc.

Free floats of various activities are calculated in column 8 of the network analysis table.

Independent float of activity i - j = F.F. – tail event slack
= F.F. – (L – E) of event *i*.

Thus independent float of activity 0 - 1 = 0 – (0 – 0) = 0,
independent float of activity 1 - 2= 0 – (2 – 2) = 0,
independent float of activity 2 - 4= 0 – (16 – 10) = – 6 ≈ 0 and so on.

Independent floats of various activities are calculated in column 9 of the table. If independent float of an activity is negative, it is taken as zero.

Interfering float of activity i - j = Max. [L.F. time of *i* - *j* – E.S. time of *j* - *k*, 0]

Thus interfering float of activity 0 - 1
= Max.[L.F. time of 0 - 1 – E.S.time of 1 - 2 or 1 - 3, 0]
= Max. [2–2, 0] = 0,
interfering float of activity 1 - 2 = Max. [L.F. time of 1 - 2 – E.S. time of 2 - 4 or 2 - 5, 0]
= Max. [16 – 10, 0] = 6, etc.

More conveniently, interfering float of an activity = T.F. – F.F.
Thus, interfering float of activity 2 - 5 = 12 – 0 = 12, etc.
Alternatively, interfering float of an activity = Head event slack.
Thus, interfering float of activity 2 - 5 = 25 – 13 = 12, etc.
Interfering floats of various activities are calculated in column 10 of the table 14.4.

EXAMPLE 14.12-4

For the network given in Fig. 14.25, determine the total, free, independent and interfering floats for each activity. Times for activities are in months.

[P.U.B.E., (T.I.T.) Nov., 2006; B.Com. April, 2007; Bombay B.Sc. (Stat.) 1984]

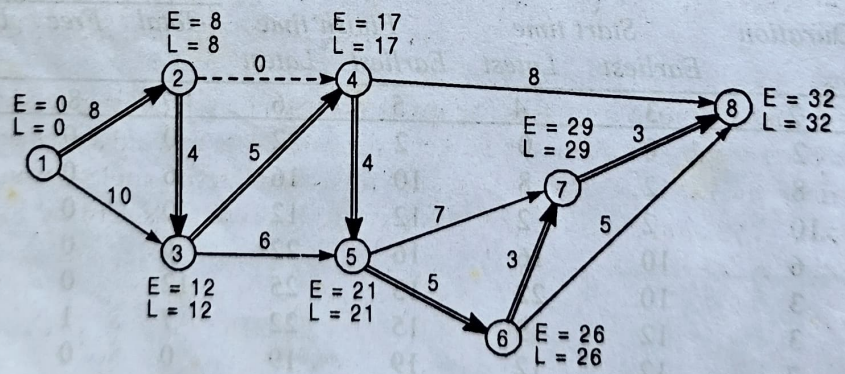


Fig. 14.25

Solution

The computations of earliest start, earliest finish, latest start and latest finish times along with floats are given in table 14.5.