

# Contents

|   |            |
|---|------------|
| <b>11 Critical Path Methods in Project Management</b> | <b>565</b> |
| 11.1 The Project Network . . . . .                    | 566        |
| 11.2 Project Scheduling . . . . .                     | 577        |
| 11.3 Exercises . . . . .                              | 586        |
| 11.4 References . . . . .                             | 592        |



# Chapter 11

## Critical Path Methods in Project Management

This is Chapter 11 of “Junior Level Web-Book *Optimization Models for decision Making*” by Katta G. Murty.

A project usually refers to an effort that is a one-time effort, one that is not undertaken on a routine production basis. For example, the construction of a skyscraper, a building, a highway, or a manufacturing facility, would be typical (civil engineering) projects. Manufacturing of large items like ships, generators, etc. would be (manufacturing) projects. In addition, the development, planning, and launching of new products; research and development programs; periodic maintenance operations; the development and installation of new management information systems; etc., are all non-routine tasks that can be considered as projects.

In this chapter we discuss techniques that help in planning, scheduling, and controlling of projects. The simplest and the most popular of these methods is the **critical path method (CPM)** which decomposes the project into a number of activities, represents the precedence relationships among them through a project network, and makes a schedule for these activities over time that minimizes the project duration, by applying on the project network the dynamic programming algorithm discussed in Chapter 10 for finding a longest route. We dis-

cuss project scheduling using CPM in this chapter.

## 11.1 The Project Network

A project is usually a collection of many individual **jobs** or **activities**. The words **job**, **activity** will be used synonymously in this chapter. The first step in CPM is to decompose the project into its constituent activities and determine the precedence relationships among them. These arise from technological constraints that require certain jobs to be completed before others can be started (for example, the job “painting the walls” can only be started after the job “erecting the walls” is completed).

We assume that each job in the project can be started and completed independently of the others within the technological sequence defined by the precedence relationships among the jobs.

If job 2 cannot be started until after job 1 has been completed, then job 1 is known as a **predecessor** or **ancestor** of job 2; and job 2 is known as a **successor** or **descendent** of job 1. If job 1 is a predecessor of job 2, and there is no other job which is a successor of job 1 and predecessor of job 2, then job 1 is known as an **immediate predecessor** of job 2, and job 2 is known as an **immediate successor** of job 1. A job may have several immediate predecessors; it can be started as soon as all its immediate predecessors have been completed. If a job has two or more immediate predecessors, by definition every pair of them must be unrelated in the sense that neither of them is a predecessor of the other.

If 1 is a predecessor of 2, and 2 is a predecessor of 3, then obviously 1 is a predecessor of 3. This property of precedence relationships is called **transitivity**. Given the set of immediate predecessors of each job, it is possible to determine the set of predecessors, or the set of successors of any job, by recursive procedures using transitivity.

The predecessor relationships are inconsistent if they require that a job has to be completed before it can be started; so, no job can be a predecessor of itself.

Because of these properties, the precedence relationships define an ordering among the jobs in a project called a **partial ordering** in

mathematics.

The planning phase of the project involves the breaking up of the project into various jobs using practical considerations, identifying the immediate predecessors of each job based on engineering and technological considerations, and estimating the time required to complete each job.

The job precedence relationship data represented in the form of a directed network called the **project network**, and the time required to complete each job, are the input data for the CPM.

Inconsistencies may appear in the predecessor lists due to human error. The predecessor data is said to be **inconsistent** if it leads to the conclusion that a job precedes itself, by the transitivity property. Inconsistency implies the existence of a circuit in the predecessor data, i.e., a subset of jobs  $1, \dots, r$ , such that  $j$  is listed as a predecessor of  $j + 1$  for  $j = 1$  to  $r - 1$ , and  $r$  is listed as a predecessor of  $1$ . Such a circuit represents a logical error and at least one link in this circuit must be wrong. As it represents a logical error, inconsistency is a serious problem.

Also, in the process of generating the immediate predecessors for an activity, an engineer may put down more than necessary and show as immediate predecessors some jobs that are in reality more distant predecessors. When this happens, the predecessor data is said to contain **redundancy**. Redundancy poses no theoretical or logical problems, but it unnecessarily increases the complexity of the network used to represent the predecessor relationships. Given the list of immediate predecessors of each job, one must always check it for any inconsistency, and redundancy, and make appropriate corrections.

As an example, we give below the precedence relationships among jobs in the project: **building a hydroelectric power station**. In this example, we have not gone into very fine detail in breaking up the project into jobs. In practice, a job like 11 (dam building) will itself be divided into many individual jobs involved in dam building.

The job duration is the estimated number of months needed to complete the job.

### Hydroelectric Power Station Building Project

| No. | Job Description   | Immediate<br>Predecessors | Job<br>Duration |
|-----|---|---------------------------|-----------------|
| 1.  | Ecological survey of dam site   |                           | 6.2             |
| 2.  | File environmental impact report<br>and get EPA approval                | 1                         | 9.1             |
| 3.  | Economic feasibility study  | 1                         | 7.3             |
| 4.  | Preliminary design and cost<br>estimation                               | 3                         | 4.2             |
| 5.  | Project approval and commitment<br>of funds                             | 2, 4                      | 10.2            |
| 6.  | Call quotations for electrical<br>equipment (turbines, generators, ...) | 5                         | 4.3             |
| 7.  | Select suppliers for electrical<br>equipment                            | 6                         | 3.1             |
| 8.  | Final design of project   | 5                         | 6.5             |
| 9.  | Select construction contractors   | 5                         | 2.7             |
| 10. | Arrange construction materials supply                                   | 8, 9                      | 5.2             |
| 11. | Dam building  | 10                        | 24.8            |
| 12. | Power station building  | 10                        | 18.4            |
| 13. | Power lines erection  | 7, 8                      | 20.3            |
| 14. | Electrical equipment installation                                       | 7, 12                     | 6.8             |
| 15. | Build up reservoir water level  | 11                        | 2.1             |
| 16. | Commission the generators   | 14, 15                    | 1.2             |
| 17. | Start supplying power   | 13, 16                    | 1.1             |

## Simple Chains in Networks, Representation Using Predecessor Node Labels

We will represent the precedence relationships among activities through a directed network. A **directed network** is a pair of sets  $(\mathcal{N}, \mathcal{A})$ , where  $\mathcal{N}$  is a set of **nodes** (also called **vertices** or **points** in the literature), and  $\mathcal{A}$  is a set of directed lines called **arcs**, each arc joining a pair of nodes.

The arc joining node  $i$  to node  $j$  is denoted by the ordered pair

$(i, j)$ ; it is **incident into**  $j$  and **incident out** of  $i$ ; node  $i$  is its **tail**, and  $j$  its **head**. For example,  $(1, 2)$  is an arc in Figure 11.2 with tail 1 and head 2.

A **chain**  $\mathcal{C}$  in the directed network  $G = (\mathcal{N}, \mathcal{A})$  from  $x_1$  (**origin** or **initial node**) to  $x_k$  (**destination** or **terminal node**) is a sequence of points and arcs alternately

$$\mathcal{C} = x_1, e_1, x_2, e_2, \dots, e_{k-1}, x_k \quad (11.1.1)$$

such that for each  $r = 1$  to  $k - 1$ ,  $e_r$  is the arc  $(x_r, x_{r+1})$ ; i.e., it is a sequence of arcs connecting the points  $x_1$  and  $x_k$ , with all the arcs directed towards the destination  $x_k$ . For example, in the directed network in Figure 11.2,  $\mathcal{C}_1 = 1, (1, 2), 2, (2, 5), 5, (5, 8), 8, (8, 13), 13, (13, 17), 17$  is a chain from node 1 to node 17 that consists of 5 arcs.

A chain is said to be a **simple chain** if no node or arc is repeated in it. The chain  $\mathcal{C}_1$  given above from node 1 to node 17 is a simple chain. A simple chain can be stored using **node labels** called **predecessor indices** or **predecessor labels**. For example, suppose the chain  $\mathcal{C}$  in (11.1.1) from  $x_1$  to  $x_k \neq x_1$  is a simple chain. The origin  $x_1$  of  $\mathcal{C}$  has no predecessors on  $\mathcal{C}$ , so its predecessor index is  $\emptyset$ . For  $2 \leq r \leq k$ ,  $x_{r-1}$  is the immediate predecessor of  $x_r$  on  $\mathcal{C}$ , hence  $x_{r-1}$  is defined to be the predecessor index of  $x_r$ .

With predecessor indices defined this way, the simple chain itself can be traced by a **backwards trace** of these predecessor indices beginning at the terminal node. From the label on the terminal node  $x_k$ , we know that the last arc on  $\mathcal{C}$ , the one incident into  $x_k$ , is  $(x_{k-1}, x_k)$ . Now go back to the predecessor node  $x_{k-1}$  of  $x_k$  on  $\mathcal{C}$ , look up the label on it, and continue in the same manner. The trace stops when the node with the  $\emptyset$  label, the origin, is reached.

As an example, in Figure 11.1 we show a simple chain from node 1 to node 14 and the predecessor labels for storing it. Each node number is entered inside the circle representing it, and its predecessor index on the simple chain is entered by its side. Nodes and arcs in the network not on this chain are omitted in this figure.

In the application discussed in this chapter, each arc in the network will have its length given, and the problem needs the longest simple chains from the origin node to every other node in the network. In the

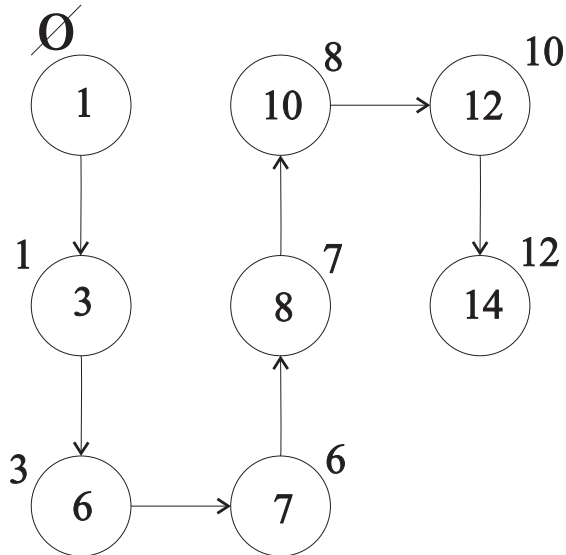


Figure 11.1: A simple chain from node 1 to node 14, and the predecessor labels on the nodes for storing it.

algorithm for solving this problem, node labels will indicate not only the predecessor indices, but also the actual lengths of the chains.

## Project Networks

We now discuss two different ways of representing the precedence relationships among the jobs in a project as a directed network. One leads to the **activity on node (AON) diagram**, and the other the **activity on arc (AOA) diagram** or **arrow diagram** of the project.

### Activity on Node (AON) Diagram of the Project

As the name implies, each job is represented by a node in this network. Let node  $i$  represent job  $i$ ,  $i = 1$  to  $n =$  number of jobs. Include arc  $(i, j)$  in the network iff job  $i$  is an immediate predecessor of job  $j$ . The resulting directed network called the **Activity on Node (AON) diagram**, is very simple to draw, but not too convenient for project



scheduling, so we will not use it in the sequel. The AON diagram of the hydroelectric power station building project is given in Figure 11.2.

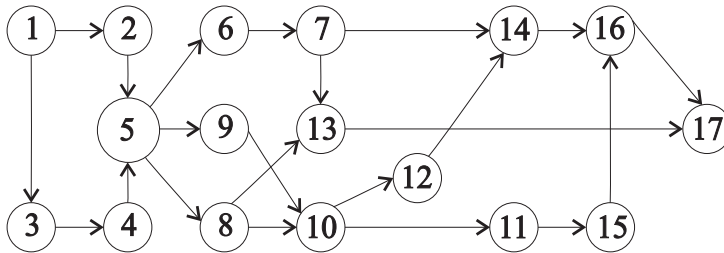


Figure 11.2: AON diagram for the hydroelectric power station building project. Jobs are represented by nodes, and  $(i, j)$  is an arc iff job  $i$  is an immediate predecessor of job  $j$ .

## Arrow Diagram of the Project

The **Arrow diagram** or the **Activity on Arc (AOA) diagram** represents jobs by arcs in the network. We refer to the job corresponding to arc  $(i, j)$  in this network, as job  $(i, j)$  itself.

**Interpretation of Nodes in the Arrow Diagram:** Nodes in the arrow diagram represent **events** over time. Node  $i$  represents the event that all jobs corresponding to arcs incident into node  $i$ , and all their predecessor jobs have been completed, and after this event any job corresponding to an arc incident out of node  $i$  can be started.

**Properties to be Satisfied by the Arrow Diagram:** The arrow diagram is drawn so as to satisfy the following properties.

**Property 1** If  $(i, j)$ ,  $(p, q)$  are two jobs, job  $(i, j)$  is a predecessor of job  $(p, q)$  iff there is a chain from node  $j$  to node  $p$  in the arrow diagram.

In order to represent the predecessor relationships through Property 1, it may be necessary to introduce **dummy arcs** which correspond

to **dummy jobs**. The need for dummy jobs is explained with illustrative examples later on. In drawing the arrow diagram, the following Property 2 must also be satisfied.

**Property 2** If  $(i, j)$ ,  $(p, q)$  are two jobs, job  $(i, j)$  is an immediate predecessor of job  $(p, q)$  iff either  $j = p$ , or there exists a chain from node  $j$  to node  $p$  in the arrow diagram consisting of dummy arcs only.

**How to Draw the Arrow Diagram for a Project:** In drawing the arrow diagram, we start with an initial node called the **start node** representing the event of starting the project, and represent each job that has no predecessor, by an arc incident out of it. In the same way, at the end we represent jobs that have no successors by arcs incident into a single final node called the **finish node** representing the event of the completion of the project.

A dummy job is needed whenever the project contains a subset  $\mathcal{A}_1$  of two or more jobs which have some, but not all, of their immediate predecessors in common. In this case we let the arcs corresponding to common immediate predecessors of jobs in  $\mathcal{A}_1$  to have the same head node and then add dummy arcs from that node to the tail node of each of the arcs corresponding to jobs in  $\mathcal{A}_1$ . As an example consider the following project, the arrow diagram corresponding to which is given in Figure 11.3.

| Job   | Immediate predecessors |
|-------|------------------------|
| $e_1$ |                        |
| $e_2$ |                        |
| $e_3$ |                        |
| $e_4$ | $e_1, e_2$             |
| $e_5$ | $e_3, e_2$             |

**A Set of Jobs With the Same Set of immediate predecessors and the Same Set of Immediate Successors:** Suppose there are  $r$  ( $\geq 2$ ) jobs, say  $1, \dots, r$ , all of which have the same set  $\mathcal{A}_1$  of immediate predecessors and the same set  $\mathcal{A}_2$  of immediate successors; and there are no other immediate successors for any of the jobs in  $\mathcal{A}_1$ , or immediate



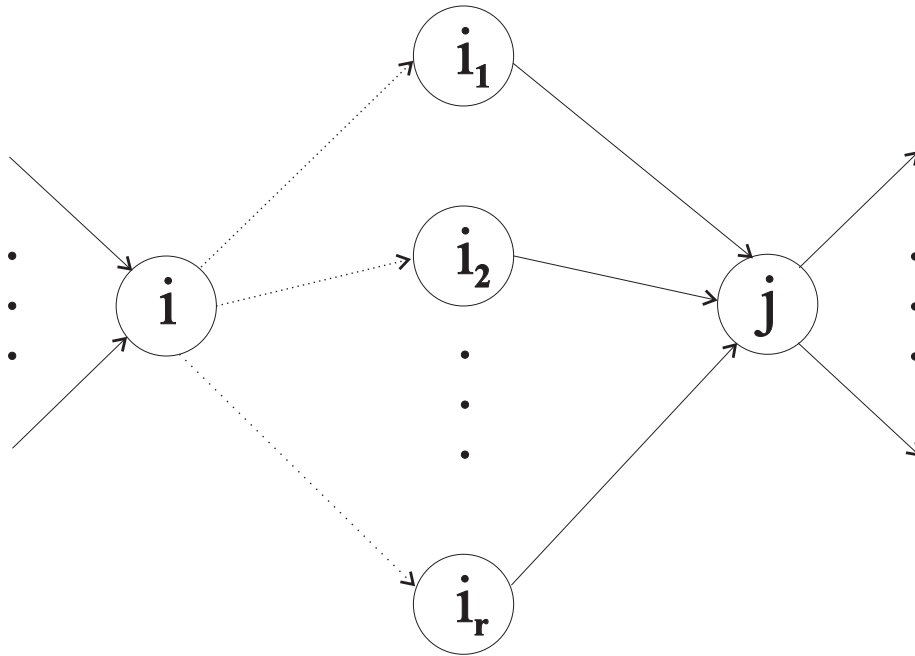


Figure 11.4: Representing jobs with identical sets of immediate predecessors and immediate successors. Arc  $(i_h, j)$  represents job  $h$ , for  $h = 1$  to  $r$ . The dashed arcs represent dummy jobs.

have identically the same set of immediate predecessors as  $b$ , introduce a new node  $q$  and represent  $b$  and each of these jobs by an arc incident out of  $q$ , and include dummy arcs  $(p_1, q), \dots, (p_r, q)$ .

If some jobs have identical sets of immediate successors, make the head node of the arcs representing these jobs the same.

We continue this way, at each stage identifying the common immediate predecessors of two or more jobs, and representing these immediate predecessors by arcs with the same head node, and letting dummy arcs issue out of this node if necessary.

In introducing dummy arcs, one should always watch out to see that precedence relationships not implied by the original data are not introduced, and those in the original specification are not omitted.

After the arrow diagram is completed this way, one can review and see whether any of the dummy arcs can be deleted by merging the two nodes on it into a single node, while still representing the predecessor relationships correctly. For example, if there is a node with a single arc incident out of it, or a single arc incident into it, and this arc is a dummy arc, then the two nodes on that dummy arc can be merged and that dummy arc eliminated. Other simple rules like these can be developed and used to remove unnecessary dummy arcs.

In this way it is possible to draw an arrow diagram for a project using simple heuristic rules. There are usually many different ways of selecting the nodes and dummy arcs for drawing the arrow diagram to portray the specified precedence relationships through Properties 1,2. Any of these that leads to an arrow diagram satisfying Properties 1,2 correctly and completely is suitable for project planning and scheduling computations. One would prefer an arrow diagram with as few nodes and dummy arcs as possible. But the problem of constructing an arrow diagram with the minimum number of dummy arcs is in general a hard problem. In practice, it is not very critical whether the number of dummy arcs is the smallest that it can be or not. Any arrow diagram obtained using the simple rules discussed above is quite reasonable and satisfactory.

As an example, the arrow diagram for the hydroelectric power plant building project discussed above is given in Figure 11.5. Here  $e_j$  is job  $j$ , with job duration entered on it. Dotted arcs are dummy jobs. Critical path (thick arcs), node labels are from CPM (Section 11.2).

Since dummy arcs have been introduced just to represent the predecessor relationships through Properties 1,2, they correspond to dummy jobs, and the time and cost required to complete any dummy job are always taken to be 0.

The transitive character of the precedence relationships, and the fact no job can precede itself, imply that an arrow diagram cannot contain any circuits (a circuit is a chain from a node back to itself); i.e., it is acyclic. As discussed in Chapter 10, an **acyclic numbering of nodes** in the arrow diagram is possible, i.e., a numbering such that if  $(i, j)$  is an arc in the network, then  $i < j$ . A procedure for numbering the nodes this way is discussed in Chapter 10. In the sequel we assume

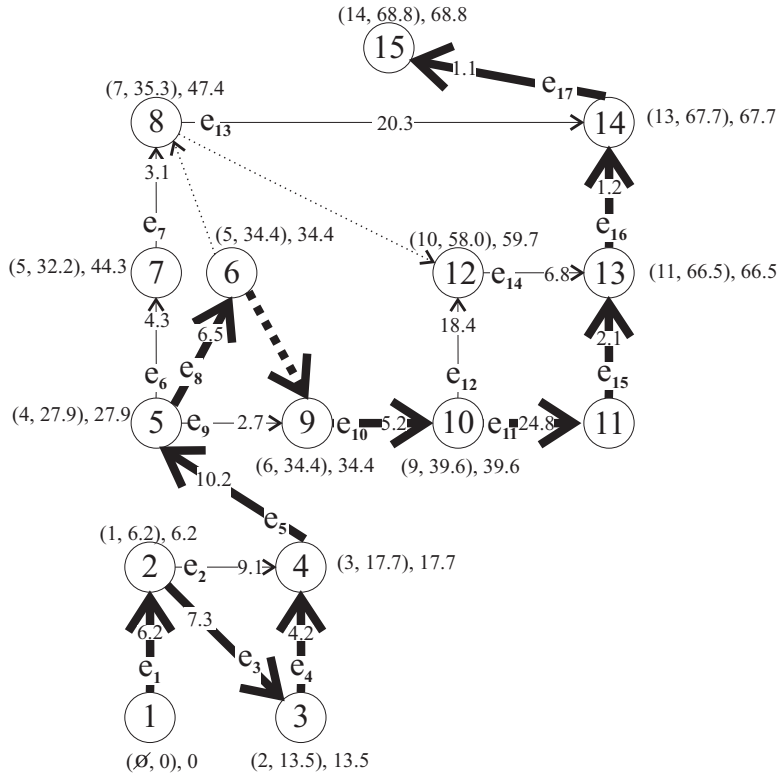


Figure 11.5: Arrow diagram for the hydroelectric power plant building problem.

that the nodes in the arrow diagram are numbered this way.

## Exercises

**1.1.1:** Given the predecessor data for a project, develop efficient procedures for checking the data for consistency and for removing redundancies in the specified immediate predecessor lists if the data is consistent.

**1.1.2:** Write a practically efficient computer program to derive an

arrow diagram for a project, given the list of immediate predecessors of each job. Include in your program simple rules to try to keep the number of nodes and the number of dummy arcs as small as possible.

## 11.2 Project Scheduling

The first step in CPM is the construction of the arrow diagram that represents the precedence relationships among the activities in the project. This is the most difficult step in CPM. It requires much thought and a very detailed analysis of the work in the project. Once this step is completed, we will have a clear understanding of what must be accomplished to complete the project successfully. This might very well be the greatest benefit of CPM.

Let  $G = (\mathcal{N}, \mathcal{A})$  with  $n$  nodes, be the arrow diagram for a project with an acyclic numbering for its nodes, and nodes 1,  $n$  as the start, finish nodes, respectively. For each job  $(i, j) \in \mathcal{A}$ , let

$$t_{ij} = \begin{array}{l} \text{the time duration required for completing job} \\ (i, j) \text{ (} t_{ij} = 0 \text{ if } (i, j) \text{ is a dummy arc)} \end{array}$$

We assume that  $t_{ij} \geq 0$  for all jobs  $(i, j)$ . Given these job durations, project scheduling deals with the problem of laying out the jobs along the time axis with the aim of minimizing the project duration. It is concerned with temporal considerations such as

- (1) how early would the event corresponding to each node materialize,
- (2) how far can an activity be delayed without causing a delay in project completion time,

etc. Make  $t_{ij}$  the length of arc  $(i, j)$  in the project network  $G$ . The minimum time needed to complete the project, known as the **minimum project duration**, is obviously the length of the longest chain from 1 to  $n$  in  $G$ ; a longest chain like that is known as a **critical path** in the arrow diagram. There may be alternate critical paths in  $G$ . Any arc which lies on a critical path is called a **critical arc**, it represents a

**critical job** or **critical activity**. Jobs which are not on any critical path are known as **slack jobs** in the arrow diagram. For each node  $i \in \mathcal{N}$  let

$$t_i = \text{the length of a longest chain from start node 1 to node } i \text{ in } G.$$

$t_n$ , the length of a critical path in  $G$ , is the minimum time duration required to complete the project. The quantity  $t_i$  is the earliest occurrence time of the event associated with node  $i$  assuming that the project has commenced at time 0. For each arc  $(i, j)$  incident out of node  $i$ ,  $t_i$  is the earliest point of time at which job  $(i, j)$  can be started after the project has commenced; hence it is known as the **early start time of job**  $(i, j)$  and denoted by  $ES(i, j)$ . For all arcs  $(i, j)$  incident out of node  $i$ ,  $ES(i, j)$  is the same, and  $t_i + t_{ij}$  is the earliest point of time that job  $(i, j)$  can be completed. This time is known as the **early finish time of job**  $(i, j)$ , and denoted by  $EF(i, j)$ . So, for all jobs  $(i, j) \in \mathcal{A}$

$$\text{Early start time for job } (i, j) = ES(i, j) = t_i$$

$$\text{Early finish time of job } (i, j) = EF(i, j) = t_i + t_{ij}$$

Since  $G$  is acyclic, the  $t_i$ s can be computed by applying the dynamic programming algorithm discussed in Chapter 10, with appropriate modifications to find the longest (instead of the shortest) chains from the origin node 1 to all the other nodes in  $G$  (instead of from every node to a fixed destination node, which was the problem discussed in Chapter 10). The process of computing the longest chains from 1 to all the other nodes in  $G$  using the recursive technique of dynamic programming is called **the forward pass** through the arrow diagram.

Once the forward pass has been completed, one schedule that gets the project completed in minimum time is to start each job at its early start time. However the forward pass identifies only one critical path, it does not identify all the critical arcs. It will be extremely helpful to the project manager if all the critical jobs can be identified, because if



a job is not critical (i.e., it is a slack job) then it can be delayed to a limited extent after its early start time without causing any delay in the whole project. And it is interesting to know how late the starting and completion of a job  $(i, j)$  can be delayed without affecting the project completion time. This informs the project management how much leeway they have in scheduling each job and still complete the project in minimum time. For job  $(i, j) \in \mathcal{A}$ , define

$$\begin{aligned} \text{LS}(i, j) &= \text{Late start time of job } (i, j) = \text{latest point of time} \\ &\quad \text{that this job can be started without affecting the} \\ &\quad \text{project completion in minimum time} \\ &= t_n - \text{length of the longest chain from node } i \text{ to} \\ &\quad \text{node } n \end{aligned}$$

$$\text{LF}(i, j) = \text{the late finish time of job } (i, j) = \text{LS}(i, j) + t_{ij}.$$

To compute the late finish times, we begin at the finish node at time point  $t_n$  and work with backwards recursion; this process is known as the **backward pass** through the arrow diagram.

An arc  $(i, j)$  is a critical arc iff  $\text{ES}(i, j) = \text{LS}(i, j)$ . Hence when both forward and backward passes have been completed, all the critical and slack arcs in the arrow diagram can be identified easily. The combined algorithm comprising the forward and backward passes is described below. In these passes  $t_{ij}$  are given data. In the forward pass, node  $i$  acquires the **forward label**  $(L_i, t_i)$  where  $t_i$  is the quantity defined above; it is the earliest event time associated with node  $i$ , and  $L_i$  is the predecessor index of node  $i$  on a longest chain from 1 to  $i$ . In the forward pass nodes are labeled in serial order from 1 to  $n$ . In the backward pass node  $i$  acquires the **backward label** denoted by  $\mu_i$ ; it is the latest event time associated with node  $i$  so that the project completion will still occur in minimum time. In the backward pass, nodes are labeled in decreasing serial order beginning with node  $n$ .

#### FORWARD PASS

**Step 1** Label the start node, node 1, with the forward label  $(\emptyset, 0)$ .

**General step**  $r$ ,  $r = 2$  to  $n$  At this stage, all the nodes  $1, \dots, r - 1$  would have been forward labeled, let these forward labels be  $(L_i, t_i)$  on node  $i = 1$  to  $r - 1$ . Find

$$t_r = \max\{t_i + t_{ir} : i \text{ is tail node on an arc incident at } r\} \quad (11.2.1)$$

Let  $L_r$  be any of the  $i$  that attains the maximum in (11.2.1). Label node  $r$  with the forward label  $(L_r, t_r)$ . If  $r = n$  go to the backward pass, otherwise go to the next step in the forward pass.

### BACKWARD PASS

**Step 1** Label the finish node, node  $n$ , with  $\mu_n = t_n$ .

**General Step**  $r$ ,  $r = 2$  to  $n$  At this stage all the nodes  $n, n - 1, \dots, n - r + 2$  would have received backward labels, let these be  $\mu_n, \dots, \mu_{n-r+2}$ , respectively. Find

$$\mu_{n-r+1} = \min\{\mu_j - t_{n-r+1,j} : j \text{ is the head node on an arc incident out of } n - r + 1\}$$

If  $r = n$  terminate; otherwise go to the next step in the backward pass.

### Discussion

As mentioned above, the forward pass is an adaptation of the dynamic programming algorithm discussed in Section 10.4, to find longest chains from node 1 to all the other nodes  $i$  in the acyclic network  $G$ . The backward pass is an adaptation of the same dynamic programming algorithm, to find longest chains from every node  $i$  in  $G$  to the finish node  $n$ , but using the fact that the longest chain from node 1 to node  $n$  has the known length  $\mu_n = t_n$ .

For any job  $(i, j) \in \mathcal{A}$ , we have

$$\begin{aligned} \text{LF}(i, j) &= \mu_j \\ \text{LS}(i, j) &= \mu_j - t_{ij} \\ \text{ES}(i, j) &= t_i \\ \text{EF}(i, j) &= t_i + t_{ij} \end{aligned}$$

The difference  $\text{LS}(i, j) - \text{ES}(i, j) = \mu_j - t_{ij} - t_i$  is known as **the total slack** or the **total float** of job  $(i, j)$  and denoted by  $\text{TS}(i, j)$ . Here is a list of some of the activity floats that are commonly used.

$$\begin{aligned} \mu_j - (t_i + t_{ij}) &= \text{total slack or total float of job } (i, j), \text{ i.e.,} \\ \text{TS}(i, j) & \\ \mu_j - t_{ij} - \mu_i &= \text{safety float of job } (i, j) \\ t_j - (t_i + t_{ij}) &= \text{free float or free slack of job } (i, j) \end{aligned}$$

Job  $(i, j)$  is a **critical job** iff  $\text{TS}(i, j) = 0$ , i.e.:

$$\text{Job } (i, j) \text{ is critical iff } \mu_j = t_i + t_{ij}$$

$$\text{Job } (i, j) \text{ is called a slack job iff } \mu_j > t_i + t_{ij}$$

$$\text{Node } i \text{ is on a critical path iff } t_i = \mu_i.$$

Hence, after the forward and backward passes, all the critical jobs are easily identified. Any chain from node 1 to  $n$  on which all the arcs are critical arcs is a **critical path**. In particular, the chain from node 1 to  $n$  traced by the forward pass labels is a critical path.

**Start Times for Critical and Slack Jobs:** Critical jobs have to start exactly at their early start times if the project has to be completed in minimum time. However, slack jobs can be started any time within the interval between their early and late start times, allowing the scheduler some freedom in choosing their starting times. One should remember that if the start time of a slack job is delayed beyond its early start time, the start times of all its successor jobs are delayed too, and this may affect their remaining total slacks.

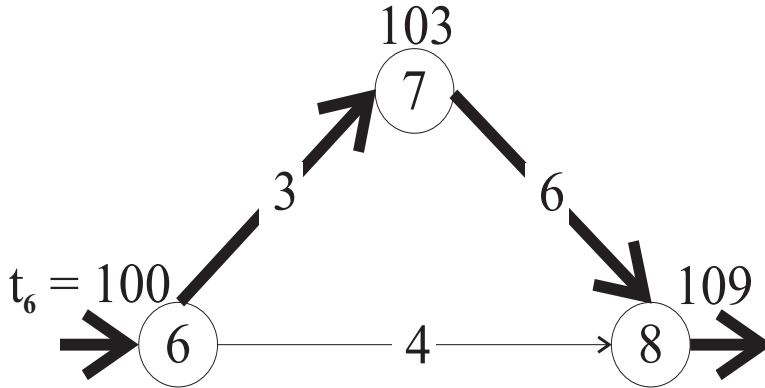


Figure 11.6: An illustration of a job (6, 8) with free slack. Thick arcs are on the critical path.

Free slack can be used effectively in project scheduling. For example, if a job has positive free slack, and its start is delayed by any amount  $\leq$  its free slack, this delay will not affect the start times or slack of succeeding jobs.

A node  $i$  is on a critical path iff  $t_i = \mu_i$ . Two nodes  $i, j$  may both be on a critical path, and yet the arc joining them  $(i, j)$  may not be a critical arc. An example is given in Figure 11.6. Here, the numbers on the arcs are the job durations, the numbers by the side of the nodes are the  $t_i$ s, and critical arcs are thick. Even though both nodes 6, 8 are on the critical path, job (6, 8) is not a critical job, and its free slack is  $109 - 100 - 4 = 5$ . Job (6, 8) has positive float even though both the nodes on it have zero slack. The start time of job (6, 8) can be anywhere between 100 to 105 time units after project start, this delay in job (6, 8) has absolutely no effect on the start times or slack of any of its successors.

## Results for the Hydroelectric Dam Building Project

Consider the arrow diagram for the hydroelectric dam building project in Figure 11.5. The critical path identified by the forward labels is marked with thick lines. For each node  $i$ , the forward pass

label, and the backward pass label  $(L_i, t_i), \mu_i$  are entered by its side. Minimum project duration is  $t_{15} = 68.8$  months. The critical path in this example is unique, as all the nodes not on it satisfy  $t_i < \mu_i$ . The ES, EF, LF, LS, TS of all the jobs listed under the project (i.e., not the dummy jobs) are given below.

The ES, EF, LF, LS, and TS  
for Jobs in the Hydroelectric  
Dam Building Project

| Job | ES   | EF   | LF   | LS   | TS   |
|-----|------|------|------|------|------|
| 1   | 0.0  | 6.2  | 6.2  | 0.0  | 0.0  |
| 2   | 6.2  | 15.3 | 17.7 | 8.6  | 2.4  |
| 3   | 6.2  | 13.5 | 13.5 | 6.2  | 0.0  |
| 4   | 13.5 | 17.7 | 17.7 | 13.5 | 0.0  |
| 5   | 17.7 | 27.9 | 27.9 | 17.7 | 0.0  |
| 6   | 27.9 | 32.2 | 44.3 | 40.0 | 12.1 |
| 7   | 32.2 | 35.3 | 47.4 | 44.3 | 12.1 |
| 8   | 27.9 | 34.4 | 34.4 | 27.9 | 0.0  |
| 9   | 27.9 | 30.6 | 34.4 | 31.7 | 3.8  |
| 10  | 34.4 | 39.6 | 39.6 | 34.4 | 0.0  |
| 11  | 39.6 | 64.4 | 64.4 | 39.6 | 0.0  |
| 12  | 39.6 | 58.0 | 59.7 | 41.3 | 1.6  |
| 13  | 35.3 | 55.6 | 67.7 | 47.4 | 12.1 |
| 14  | 58.0 | 64.8 | 66.5 | 59.7 | 1.7  |
| 15  | 64.4 | 66.5 | 66.5 | 64.4 | 0.0  |
| 16  | 66.5 | 67.7 | 67.7 | 66.5 | 0.0  |
| 17  | 67.7 | 68.8 | 68.8 | 67.7 | 0.0  |

We will now explain how to interpret these results. For example, consider job 3 (corresponding to arc  $e_3 = (2, 3)$  in the arrow diagram in Figure 11.5) in this project. Its early start time is  $t_2 = 6.2$  months. That means that the earliest time at which this job can be started (this is the time by which all its predecessor jobs would have been completed) is 6.2 months after project commencement. This is the kind of input that project managers need, since it provides information on when to

order any special equipment or trained personnel needed to carry out this job, to arrive at project site. And the late start time of this job is  $\mu_3 - t_{23} = 6.2$  months, same as its early start time. This means that if job 3 is not started at 6.2 months time after project commencement, the completion of the whole project will be delayed beyond its minimum duration of 68.8 months. Since the early and late start times of job 3 are equal, it is a critical job. In the same way, jobs 1, 4, 5, 8, 10, 11, 15, 16, 17 are also critical jobs, and a similar interpretation can be given to their early start times.

Consider job 2 (corresponding to arc  $e_2 = (2, 4)$  in the arrow diagram in Figure 11.5) in this project. Its early and late start times are 6.2 and 8.6 months. Since its late start time is  $>$  than its early start time, this job is a slack job. It can be started anytime after 6.2 months (after project commencement); but unless it is started before 8.6 months, project completion will be delayed. The free slack of this job  $(2, 4)$  is  $t_4 - t_2 - t_{24} = 17.7 - 6.2 - 9.1 = 2.4$ . It implies that starting this job any time between 6.2 to 8.6 months after project commencement, has no effect on the early or late start of any succeeding job.

Consider job 12 corresponding to arc  $(10, 12)$  in the arrow diagram in Figure 11.5. Its early and late start times are 39.6 and 41.3 months respectively, but its free slack is  $t_{12} - t_{10} - t_{10,12} = 58.0 - 39.6 - 18.4 = 0$ . It implies that this job can be started any time after 39.6 months (after project commencement), but unless it is started before 41.3 months, the whole project will be delayed. And since its free slack is 0, if it is started some time after 39.6 but before 41.3 months, the early start times of succeeding jobs will be affected (it can be verified that the early start time of job 14 will change depending on the start time of job 12 between 39.6 to 41.3 months).

In the same way, the output from the forward and backward passes of the above algorithm provides extremely useful planning information to the project manager for scheduling the various jobs over time and in evaluating the effects of any unavoidable changes in the schedule on the project completion date.

## Summary

In this chapter, we discussed how to represent the precedence relationships among the jobs in a project using a directed project network. Given this project network, and the time durations of the various jobs, we discussed a method for scheduling the jobs over time to complete the project in minimum time. The method is based on the dynamic programming algorithm of Section 10.4 for finding optimal routes in acyclic networks. This is the most basic critical path method, and serves our purpose of exposing the reader to elementary but very important optimization tools for project management.

Sometimes, it may be necessary to complete a project earlier than the minimum duration for it as determined by normal job durations. In this case it will be necessary to complete some jobs in time less than their normal duration, by allowing workers to work overtime, etc. Given the unit cost of expediting each job, there is an algorithm called **project shortening cost minimization algorithm**, which determines the subset of jobs to be expedited, and each by how much, in order to complete the project within the desired duration at minimum shortening cost. Since project managers are often under pressure to complete projects early, this is a very useful algorithm for them. For a discussion of this algorithm, see for example [K. G. Murty, 1992].

## Resource Constrained Project Scheduling

So far we assumed that the only constraints in scheduling jobs over time are those imposed by the predecessor relationships among them. When this assumption is valid, project scheduling becomes a very simple problem for which we discussed very efficient algorithms. This provides a basic introduction to project scheduling.

However, real world project scheduling usually involves many other constraints. To carry out jobs in practical project scheduling problems, we require resources such as a crane, or other piece of equipment, or trained personnel, etc. Two or more jobs may require the same resources, and it may not be possible to carry them out simultaneously because of limited supply of resources, even though the precedence constraints do not prevent them from being scheduled simultaneously. The limited availability of resources imposes a new set of constraints.

Before starting a job, the project scheduler now has to make sure that all its predecessors have been completed, and also that the resources required to carry it out are available. Problems of this type are known as **resource constrained project scheduling problems**.

Practical resource constrained project scheduling normally leads to very large combinatorial optimization problems, for which efficient exact algorithms are not known at the moment. Hence, a variety of heuristic algorithms have been developed for resource constrained project scheduling, discussion of these techniques is beyond the scope of this book. Some of the advanced books that the interested reader can see are: A. Battersby [1967], P. J. Burman [1972], S. Elmaghraby [1977], J. D. Weist and F. K. Levy [1977], and R. J. Willis and N. A. J. Hastings [1976].

Since the 1960s, the network-based CPM discussed in this chapter has become a part of the language of project management, and has been used extensively to provide very basic information for planning, scheduling and controlling large projects. The glamorous successes claimed for their initial applications, and the adoption of these models as standard requirement in contracts by many governments at that time, have added to their importance. Computer packages for these network based techniques specialized to the needs of a variety of industries continue to be the best sellers of all optimization software.

### 11.3 Exercises

**11.1:** A project consists of jobs A to L with immediate predecessor (IP) and job duration data as given below.

|          |    |   |   |     |   |   |     |   |     |       |   |   |
|----------|----|---|---|-----|---|---|-----|---|-----|-------|---|---|
| Activity | A  | B | C | D   | E | F | G   | H | I   | J     | K | L |
| IPs      |    |   |   | A,B | B | B | C,F | B | E,H | C,D,F | J | K |
| Duration | 13 | 8 | 9 | 10  | 6 | 5 | 4   | 7 | 3   | 4     | 8 | 5 |

Draw the arrow diagram, and prepare a schedule for the jobs that minimizes the project duration.

**11.2: A new product development project:** The following is



the list of activities involved in developing a new product at a company. The expected duration of each activity, in weeks, is also given.

Read the list of activities very carefully, and using your engineering knowledge and judgment, write down what you think are the immediate predecessors of each activity. Justify your choice carefully. Using this information, draw the arrow diagram for the project and prepare a time schedule for the activities to complete the project in minimum time.

| No. | Activity                  | Duration<br>(weeks) | Immediate<br>predecessors |
|-----|---------------------------|---------------------|---------------------------|
| 1   | Generate marketing plans  | 3                   |                           |
| 2   | Assign responsibilities   | 1                   |                           |
| 3   | Consolidate plans         | 1                   |                           |
| 4   | Review product lines      | 3                   |                           |
| 5   | Hire prototype artist     | 3                   |                           |
| 6   | Design prototypes         | 7                   |                           |
| 7   | Hire layout artist        | 2                   |                           |
| 8   | Hire new production crew  | 4                   |                           |
| 9   | Train new production crew | 7                   |                           |
| 10  | Review prototypes         | 1                   |                           |
| 11  | Final selection           | 4                   |                           |
| 12  | Prepare national ads.     | 5                   |                           |
| 13  | Approve advertising       | 1                   |                           |
| 14  | Produce advertising       | 7                   |                           |
| 15  | Draft press releases      | 2                   |                           |
| 16  | Press ready               | 1                   |                           |

([H. J. Thamhain, 1992])

**11.3: Coke Depot Project:** A depot is to be built to store coke and to load and dispatch trucks. There will be three storage hoppers (SH. in abbreviation), a block of bunkers (B. in abbreviation), interconnecting conveyers (abbreviated as C.), and weigh bridges (called WB.). Around the bunkers there will be an area of hard-standing and an access road will have to be laid to the site. Data on this project, and the duration (in weeks) of each job are given below. Draw an arrow diagram for this project, and determine the earliest and latest start

and finish times, and the total float of each job. Schedule the jobs so that the project is finished as quickly as possible. (R. J. Willis and N. A. J. Hastings [1976])

| No. | Job                              | IPs    | Duration |
|-----|----------------------------------|--------|----------|
| 1.  | B. piling                        |        | 5        |
| 2.  | Clear site for SH.               |        | 8        |
| 3.  | B. excavation for cols.          | 1      | 4        |
| 4.  | SH. excavations for C.           | 2, 3   | 4        |
| 5.  | Concrete tops of piles for B.    | 3      | 3        |
| 6.  | Place cols. for B.               | 5      | 4        |
| 7.  | Excavate access road             | 5      | 4        |
| 8.  | Put in B.                        | 6      | 3        |
| 9.  | Stairways inside B.              | 6      | 1        |
| 10. | Excavate pit for WB.             | 4      | 6        |
| 11. | Concrete for SH.                 | 4      | 12       |
| 12. | Main C. foundation               | 4      | 4        |
| 13. | Brick walls for B.               | 8, 9   | 3        |
| 14. | Clad in steel for B.             | 8, 9   | 1        |
| 15. | Install internal equip. in B.    | 8, 9   | 6        |
| 16. | Erect gantry for main C.         | 12, 6  | 1        |
| 17. | Install C. under hoppers         | 11     | 1        |
| 18. | Concrete pit for WB.             | 11, 10 | 2        |
| 19. | Excavate for hard-standing       | 7      | 9        |
| 20. | Lay access roadway               | 7      | 9        |
| 21. | Install outloading equip. for B. | 15     | 2        |
| 22. | Line B.                          | 13, 14 | 1        |
| 23. | Install main C.                  | 16     | 1        |
| 24. | Build weighhouse                 | 18     | 4        |
| 25. | Erect perimeter fence            | 19     | 4        |
| 26. | Install C. to SH.                | 17, 23 | 1        |
| 27. | Install WB.                      | 24     | 1        |
| 28. | Lay hard-standing                | 19, 18 | 6        |
| 29. | Commission hoppers               | 26     | 1        |

IP = Immediate predecessors

**11.4:** Draw the arrow diagram and determine the early and late start and finish times of the various jobs in the following project (A. Kanda and N. Singh [1988]).

| Project: Setting Up a Fossil Fuel Power Plant |                                      |        |                       |
|---|--------------------------------------|--------|-----------------------|
| No.   | Job                                  | IPs    | Job duration (months) |
| 1.  | Land acquisition                     |        | 6                     |
| 2.  | Identi. trained personnel            | 1      | 3                     |
| 3.  | Land dev. & infrastructure           | 1      | 2                     |
| 4.  | Control room eng.                    | 1      | 12                    |
| 5.  | Lag in turbine civil works           | 1      | 8                     |
| 6.  | Delivery of TG                       | 1      | 12                    |
| 7.  | Delivery of boiler                   | 1      | 10                    |
| 8.  | Joining time for personnel           | 2      | 3                     |
| 9.  | Boiler prel. civil works             | 3      | 2                     |
| 10.   | Control room civil works             | 4      | 5                     |
| 11.   | TG civil works                       | 5      | 9                     |
| 12.   | Training                             | 8      | 6                     |
| 13.   | Boiler final civil works             | 9      | 9                     |
| 14.   | Erection of control room             | 10     | 8                     |
| 15.   | Erection of TG                       | 6, 11  | 10                    |
| 16.   | Boiler erection                      | 7, 13  | 12                    |
| 17.   | Hydraulic test                       | 16     | 2                     |
| 18.   | Boiler light up                      | 14, 17 | 1.5                   |
| 19.   | Box up of turbine                    | 15     | 3                     |
| 20.   | Steam blowing, safety valve floating | 18, 19 | 2.5                   |
| 21.   | Turbine rolling                      | 20     | 1.5                   |
| 22.   | Trial run                            | 21     | 1                     |
| 23.   | Synchronization                      | 22     | 1                     |

IP = Immediate predecessors

**11.5:** Draw an arrow diagram for each of the following projects. Prepare a schedule for the various jobs in each project to complete it in minimum time. (R. Visweswara Rao).

(a) Data Process and Collection System Design for a Power Plant

| No. | Activity                      | IPs | Duration (days) |
|-----|-------------------------------|-----|-----------------|
| 1.  | Prel. Syst. description       |     | 40              |
| 2.  | Develop specs.                | 1   | 100             |
| 3.  | Client approval & place order | 2   | 50              |
| 4.  | Develop I/O summary           | 2   | 60              |
| 5.  | Develop alarm list            | 4   | 40              |

contd.

| Data Process and Collection System Design contd. |                                  |            |                    |
|--|----------------------------------|------------|--------------------|
| No.  | Activity                         | IPs        | Duration<br>(days) |
| 6.   | Develop log<br>formats           | 3, 5       | 40                 |
| 7.   | Software def.                    | 3          | 35                 |
| 8.   | Hardware<br>requirements         | 3          | 35                 |
| 9.   | Finalize I/O<br>summary          | 5, 6       | 60                 |
| 10.  | Anal. performance<br>calculation | 9          | 70                 |
| 11.  | Auto. turbine<br>startup anal.   | 9          | 60                 |
| 12.  | Boiler guides anal.              | 9          | 30                 |
| 13.  | Fabricate & ship                 | 10, 11, 12 | 400                |
| 14.  | Software preparation             | 7, 10, 11  | 80                 |
| 15.  | Install & check                  | 13, 14     | 130                |
| 16.  | Termination &<br>wiring lists    | 9          | 30                 |
| 17.  | Schematic wiring<br>lists        | 16         | 60                 |
| 18.  | Pulling & term.<br>of cables     | 15, 17     | 60                 |
| 19.  | Operational test                 | 18         | 125                |
| 20.  | First firing                     | 19         | 1                  |

IP = Immediate predecessors

## (b) Sewer and Waste System Design for a Power Plant

| No. | Activity                             | IPs | Duration<br>(days) |
|-----|--------------------------------------|-----|--------------------|
| 1.  | Collection system<br>outline         |     | 40                 |
| 2.  | Final design &<br>approval           | 1   | 30                 |
| 3.  | Issue construction<br>drawings       | 2   | 30                 |
| 4.  | Get sewer pipe<br>& manholes         | 1   | 145                |
| 5.  | Fabricate & ship                     | 3,4 | 45                 |
| 6.  | Treat. system<br>drawings & approval |     | 70                 |

IP = Immediate predecessors, Contd.

## (b) Sewer and Waste System Design for a Power Plant

| No. | Activity                                     | IPs  | Duration<br>(days) |
|-----|--|------|--------------------|
| 7.  | Issue treat. system<br>construction drawings | 6    | 30                 |
| 8.  | Award contract                               | 7    | 60                 |
| 9.  | Final construction                           | 8, 5 | 300                |

IP = Immediate predecessors

## (c) Electrical Auxiliary System Design for a Nuclear Plant

| No. | Activity                                  | IPs        | Duration<br>(days) |
|-----|---|------------|--------------------|
| 1.  | Aux. load list                            |            | 120                |
| 2.  | 13.8 switchgear<br>load ident.            | 1          | 190                |
| 3.  | 4.16kv & 480 v. switchgear<br>load ident. | 1          | 45                 |
| 4.  | Vital AC load<br>determination            | 1          | 300                |
| 5.  | DC load determ.                           | 1          | 165                |
| 6.  | Voltage drop study                        | 2          | 84                 |
| 7.  | Diesel gen. sizing                        | 3          | 77                 |
| 8.  | Inventer sizing                           | 4          | 20                 |
| 9.  | Battery sizing                            | 5, 8       | 40                 |
| 10. | DC fault study                            | 9          | 80                 |
| 11. | Prel. AC fault<br>current study           | 6, 7       | 20                 |
| 12. | Power transformer<br>sizing               | 2, 11      | 80                 |
| 13. | Composite oneline<br>diagram              | 2,3        | 72                 |
| 14. | Safety (class 1E)<br>system design        | 13         | 200                |
| 15. | Non-class 1E<br>system design             | 13         | 190                |
| 16. | Relaying oneline<br>& metering dia.       | 13         | 80                 |
| 17. | 3-line diagram                            | 14, 15, 16 | 150                |
| 18. | Synchronizing &<br>phasing diagrams       | 17         | 100                |
| 19. | Client review                             | 10, 18     | 25                 |
| 20. | Equipment purchase<br>& installation      | 19         | 800                |

## 11.4 References

- A. BATTERSBY, 1967, *Network Analysis for Planning and Scheduling*, Macmillan & Co., London.
- P. J. BURMAN, 1972, *Precedence Networks for Project Planning and Control*, McGraw-Hill, London.
- D. DIMSDALE, March 1963, "Computer Construction of Minimal Project Network," *IBM Systems Journal*, 2(24-36).
- S. E. ELMAGHRABY, 1977, *Activity Networks*, Wiley, NY.
- A. C. FISHER, D. S. LIEBMAN, and G. L. NEMHAUSER, July 1968, "Computer Construction of Project Networks," *Communications of the Association for Computing Machinery*, 11(493-497).
- A. KANDA and V. R. K. RAO, May 1984, "A Network Flow Procedure for Project Crashing with Penalty Nodes," *European Journal of Operational Research*, 16, no. 2(123 -136).
- A. KANDA and N. SINGH, July 1988, "Project Crashing with Variations in Reward and Penalty Functions: Some Mathematical Programming Formulations," *Engineering Optimization*, 13, no. 4(307-315).
- J. E. KELLY, Jr., 1961, "Critical Path Planning and Scheduling: Mathematical Basis," *Operations Research*, 9(296-320).
- J. E. KELLY, Jr. and M. R. WALKER, Dec. 1959, "Critical Path Planning and Scheduling," *Proceedings of the Eastern Joint Computer Conference*, Boston, MA.
- H. J. THAMHAIN, 1992, *Engineering Management Managing Effectively in Technology-Based Organizations*, Wiley-Interscience, NY.
- J. D. WEIST and F. K. LEVY, 1977, *A Management Guide to PERT/CPM*, Prentice-Hall, Englewood Cliffs, NJ, 2nd Ed.
- R. J. WILLIS and N. A. J. HASTINGS, 1976, "Project Scheduling With Resource Constraints Using Branch and Bound Methods," *Operations Research Quarterly*, 27, no. 2, i(341-349).

# Index

For each index entry we provide the section number where it is defined or discussed first.

## **AON diagram 11.1**

- Activities 11.1
- Ancestor 11.1
- Arcs 11.1
- Arrow diagram 11.1

## **Backward pass 11.2**

- Backward trace 11.1

## **Chain 11.1**

- Simple 11.1
- Critical 11.2
  - Activity 11.2
  - Arc 11.2
  - Job 11.2
- Critical path 11.2
  - Algorithm to find 11.2
  - Method (CPM) 11.2

## **Descendent 11.1**

- Directed network 11.1

## **Early finish 11.2**

- Early start 11.2

## **Forward pass 11.2**

- Free float 11.2
- Free slack 11.2

## **Head 11.1**

## **Jobs 11.1**

## **Node 11.1**

## **Partial ordering 11.1**

- Predecessor 11.1
  - Indices 11.1
  - Immediate 11.1
  - Labels 11.1
- Project network 11.1
- Project scheduling 11.2

## **Redundancy 11.1**

- Resource constraints 11.2

## **Slack jobs 11.2**

- Successor 11.1
  - Immediate 11.1

## **Total float 11.2**

- Total slack 11.2
- Transitivity 11.1