Contents

7 Critical Path Methods **544**

7.1 Project Scheduling 553

7.2 The Project Shortening Cost Curve 559

7.3 Resource Constrained Project Scheduling Problems . . 572

7.4 PERT 573

7.5 Exercises 574

7.6 References 584

Chapter 7

Critical Path Methods in Project Networks

Critical path methods (CPM in abbreviation) deal with the application of shortest chain and minimum cost flow algorithms to schedule the jobs in a project along the time axis. Large civil engineering projects (construction of a skyscraper, a highway, etc.); projects that involve the manufacture of large items like ships, generators; projects that involve the development, planning, and launching of new products; large scale research and development projects; etc.; consist of a collection of many individual **jobs** or **activities** with a **partial ordering** defined among them, which arises 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). The words *job, activity* are used synonymously in this chapter. We assume that each job in the project can be started and completed independently of the others within the technological sequence defined by the partial ordering among the jobs. The partial ordering among the jobs defines a directed network known as the **project network**. The simplest CPM derives the project duration and a schedule for the various jobs to achieve this duration, given the project network and the time needed to complete each job. Another critical path model takes as input the cost of applying more workers or other resources to shorten the job durations, and determines which jobs need to be expedited to achieve a specified project duration at minimum cost. In this sense, CPM are concerned with obtaining the trade-off between cost and duration of the project. CPM are most useful in projects such as construction projects, for which there has been considerable experience, and a data base is available to derive reliable cost estimates. CPM are among the most commonly used optimization techniques. Software based on CPM for project planning, analysis, scheduling, and control, is one of the biggest money-makers among all OR software.

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.

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, . . ., 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. This is followed by a discussion of two different ways of representing the precedence relationships among the jobs in a project as a directed network.

Hydroelectric Power Station Building Project

No.	Job Description	Immediate Predecessors	Job Duration
1.	Ecological survey of dam site		6.2
2.	File environmental impact report and get EPA approval	1	9.1
3.	Economic feasibility study	1	7.3
4.	Preliminary design and cost estimation	3	4.2
5.	Project approval and commitment of funds	2, 4	10.2
6.	Call quotations for electrical equipment (turbines, generators, ...)	5	4.3
7.	Select suppliers for electrical 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

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 7.1.

Arrow Diagram of the Project

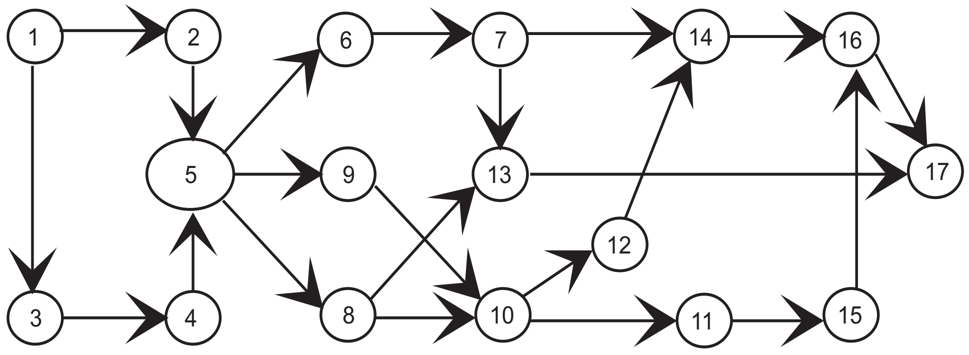

Figure 7.1: AON diagram for the hydroelectric power station building project.

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. Nodes in the arrow diagram represent **events** over time. Node i represents the event that all jobs corresponding to arcs incident into node i have been completed, and after this event any job corresponding to an arc incident out of node i can be started. The arrow diagram is drawn so as to satisfy the following property.

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.

In drawing the arrow diagram, we start with an initial node called the *start node* representing the event of starting the project, and

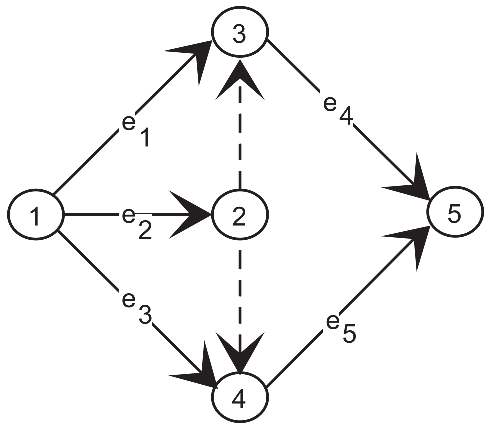

Figure 7.2: Arrow diagram. Dashed arcs represent dummy jobs.

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 7.2.

Job	Immediate predecessors
e_1	
e_2	
e_3	
e_4	e_1, e_2
e_5	e_3, e_2

Suppose there are r ($\geqq 2$) jobs, say $1, \ldots, r$, all of which have the same set $\mathcal{A}_1$ of immediate predecessors and the same set $\mathcal{A}_2$ of

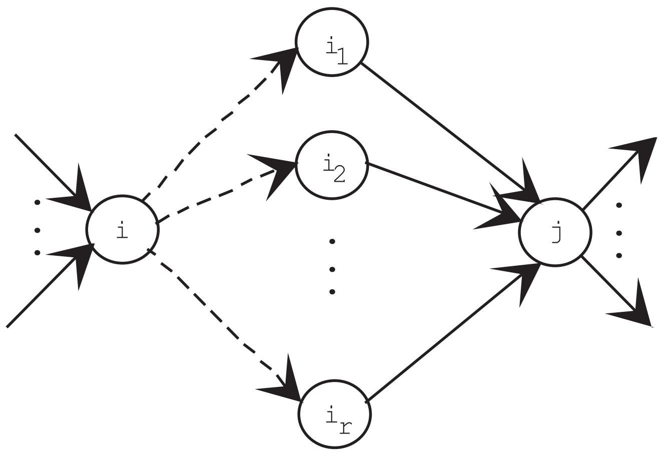

Figure 7.3: Representing jobs with identical sets of immediate predecessors and immediate successors. Arc (i_h, j) represents job h, for $h =$ 1 to r. The dashed arcs represent dummy jobs.

immediate successors; and there are no other immediate successors for any of the jobs in $\mathcal{A}_1$, or immediate predecessors for any of the jobs in $\mathcal{A}_2$. Then, all jobs in the set $\mathcal{A}_1$ can be represented by arcs incident into a common node, i, say; and all jobs in the set $\mathcal{A}_2$ can be represented by arcs incident out of a common node j, say. Then the jobs 1,..., r, can be represented by r parallel arcs joining nodes i, j. However project engineers do not usually like to deal with parallel arcs, so we introduce additional nodes $i_1, \ldots, i_r$ and represent job h by the arc $(i_h, j), h = 1$ to r; and include dummy arcs (i, i_h) for each $h = 1$ to r. See Figure 7.3.

If a job b has a single immediate predecessor a, then b can be represented by an arc incident out of the head node of the arc representing a.

If job b has more than one immediate predecessor, let $p_1, \ldots, p_r$ be the head nodes of all the arcs representing its immediate predecessors. If no other job has the same set of immediate predecessors, see if it is possible to represent b by an arc incident out of one of the nodes $p_1, \ldots, p_r$ with dummy arcs emanating from the other nodes in this set into that node. If this is not possible, or if there are other jobs which 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 pred- ecessors 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. For example, a procedure is described in Exercise 7. 2 for converting the AON diagram into an arrow diagram. However, the resulting arrow diagram has too many nodes and dummy arcs and hence it is not efficient to use it. 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 (see Krishnamoorthy and Deo [1979], and Exercise 7.11). In practice, it is not very critical whether the number of dummy arcs is the smallest that it can be or not. Any arrow

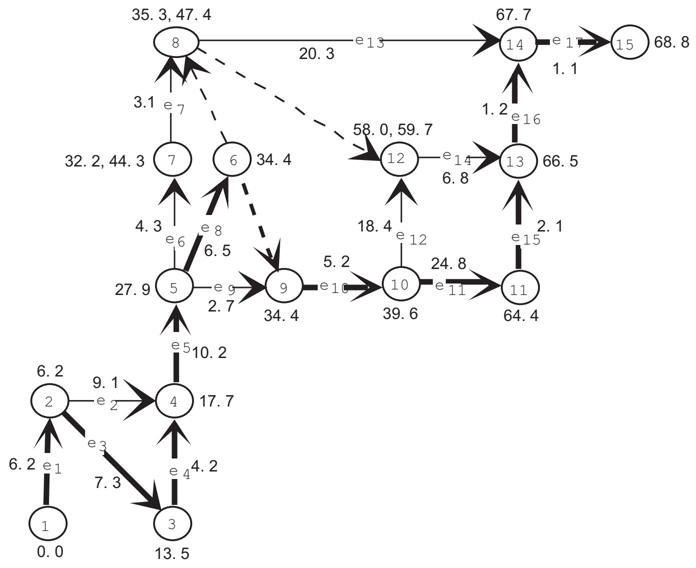

Figure 7.4: Arrow diagram for hydroelectric power plant building problem. Numbers by side of job arcs are job durations. Critical path is marked with thick arcs. If earliest and latest occurrence times for a node are the same, that value is entered by the side of the node, otherwise they are both entered in this order.

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 7.4.

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 (i.e., it is acyclic). By the results in Chapter 1, 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 $i < j$. In the sequel we assume that the nodes in the arrow diagram are numbered this way.

Exercises

7.1 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.

7.2 Let G be the AON diagram for a project. Replace each node i in G by an arc of the form (i', i''). Let G$'$ be the resulting network. In G$'$ let the arc (i', i'') represent the same job that node i represented in G. Also, let all arcs in G$'$ which correspond to arcs in G be dummy arcs. Show that G$'$ is an arrow diagram for the project.

7.3 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.

7.1 Project Scheduling

Let G $= (\mathcal{N}, \mathcal{A})$, with $|\mathcal{N}| = n$, be the arrow diagram for a project with an acyclic numbering for its nodes, and nodes 1, n as the start, finish nodes, respectively. Given the 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. For $(i, j) \in \mathcal{A}$ let $t_{ij} \geqq 0$ be the time duration required for completing job (i, j) ($t_{ij} = 0$ if (i, j) is a dummy arc), make t_{ij} the length of arc (i, j) in 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 $i \in \mathcal{N}$ let t_i denote the length of a longest chain from start node 1 to node i 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).

Since G is acyclic, the t_is can be computed by applying the algorithm discussed in Section 4.4, with appropriate modifications to find the longest instead of the shortest chain, on G, this process 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. If the duration of any critical job increases by ϵ while all the other data remain unchanged, the project duration also increases by ϵ. If it is required to complete the project in less than t_n units of time, it is necessary to reduce the time required to complete at least one job on every critical path. For this it is helpful if all the critical arcs can be identified. The forward pass identifies only one critical path, it does not identify all the critical arcs.

Slack jobs can be delayed to a limited extent without causing any delay in the whole project. 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. We define the **late start time of job** (i, j) to be the latest time that this job can be started without affecting the project completion in minimum time and denote it by LS(i, j). The **late finish time of job** (i, j), denoted by LF(i, j) is LS(i, j) + t_{ij}. To compute the late finish times, we begin at the finish node at time point t_n and work backwards, this process is known as the **backward pass** through the arrow diagram. An arc (i, j) is a critical arc iff ES(i, j) = 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 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 μ_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 $1, \ldots, r-1$ would have been forward labeled, let these forward labels be (L_i, t_i) on node $i = 1$ to $r - 1$. Find

$$t_r = \text{ Maximum } \{t_i + t_{ir} : i \in \mathbf{B}_r\} \tag{7.1}$$

where $\mathbf{B}_r$ is the set of tail nodes on arcs incident into node r. Let L_r be any of the i that attains the maximum in (7.1). 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

$$\mu_{n-r+1} = \text{ Minimum } \{\mu_j - t_{n-r+1,j} : j \in \mathbf{A}_{n-r+1}\} \qquad (7.2)$$

where $\mathbf{A}_{n-r+1}$ is the set of head nodes on arcs incident out of $n - r + 1$. If $r = n$ terminate; otherwise go to the next step in the backward pass.

Discussion

The fact that t_i in the forward label on node i is the length of the longest chain from node 1 to i follows from the results in Section 4.4.

We will now show that the backward label, μ_i, on node i is the latest point of time at which the event associated with node i has to occur if the project is to be completed in minimum time. This is clearly true for $i = n$. Set up an induction hypothesis that this statement is true for $i \geqq n - r + 2$ for some r between 2 and n. Suppose the minimum in (7.2) is attained by $j = p$. So, $(n - r + 1, p) \in \mathcal{A}$ (this implies that $p \geqq n - r + 2$), and μ_{n-r+1} defined in (7.2) satisfies $\mu_{n-r+1} + t_{n-r+1,p} = \mu_p$. Thus, if the event associated with node $(n - r + 1)$ does not occur before μ_{n-r+1}, then the event associated with node p cannot occur before μ_p, and since $p \geqq n - r + 2$, by the induction hypothesis this implies that the project will not be completed in minimum time. Also, from (7.2) it is clear that if the event associated with node $(n - r + 1)$ occurs at time μ_{n-r+1}, then the events associated with nodes $i \geqq n - r + 1$ can occur by time μ_i. All these facts together imply that under the induction hypothesis, the statement made at the beginning must also be true for $i = n - r + 1$. By induction, it is true for all i.

Hence, for any $(i, j) \in \mathcal{A}$, LF$(i, j) = \mu_j$, and so LS$(i, j) = \mu_i - t_{ij}$. We have already seen that ES$(i, j) = t_i$. The difference LS(i, j) - 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 TS(i, j). Also, the following activity floats can be similarly interpreted.

$\mu_j - \mu_i - t_{ij} =$ **Safety float** of job (i, j)

$t_j - t_i - t_{ij} =$ **free float** *or* **free slack** of job (i, j)

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

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

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.

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

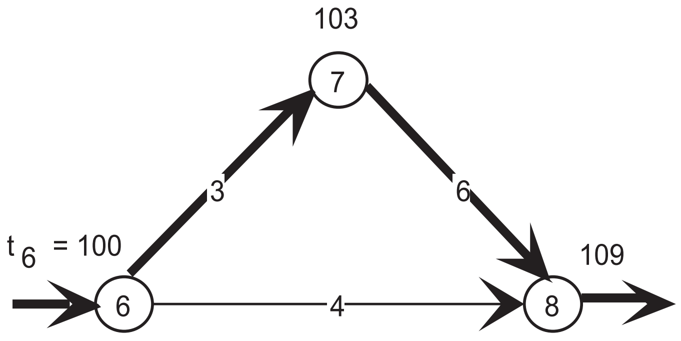

Figure 7.5: An illustration of a job (6, 8) with free slack. Thick arcs are on critical path.

amount $\leqq$ 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 7.5. Here, the numbers on the arcs are the job durations, the numbers by the side of the nodes are the t_is, 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.

Consider the arrow diagram for the hydroelectric dam building project in Figure 7.4. The critical path identified by the forward labels is marked with thick lines. For each i, if $t_i = \mu_i$ we entered their common value by the side of node i or entered the pair t_i, μ_i in that order if they are not equal. Minimum project duration is 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 above. It can be verified that job 2 has positive free slack of 2.4 in this example.

7.2 The Project Shortening Cost Curve

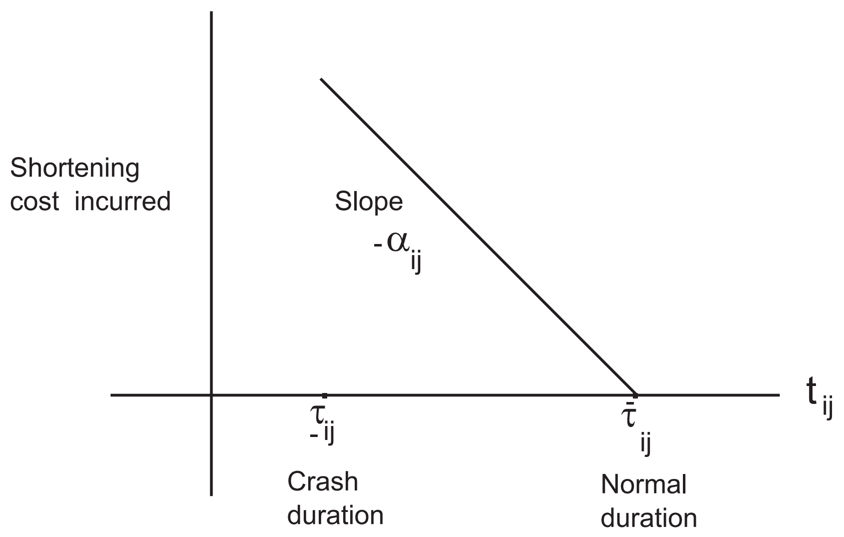

Figure 7.6: Cost of shortening job (i, j)

Let G = $(\mathcal{N}, \mathcal{A})$ be the arrow diagram for a project with 1, n as the start and finish nodes. Suppose it is required to complete a project before the minimum completion time, t_n, computed as discussed above, to meet a desired project due date. Then, ways have to be found to shorten the project duration by expediting one or more jobs. In practice, most job durations can be reduced by devoting additional resources (such as more workers, machines, overtime, etc.). This operation of expediting is known as **job shortening** *or* **crashing**, and the expense incurred on it is known as the **job shortening cost**. We consider here the problem of determining which subset of jobs to shorten, and each by how much, in order to complete the project by the given due date at minimum shortening cost. When a job is shortened, the job shortening cost is incurred in addition to the normal cost of carrying out the job. The normal cost is incurred anyway whether the job is shortened or not, hence we ignore it here, and try to minimize the total job shortening cost. We assume that the following data is available for each job (i, j) in the project.

$\overline{\tau}_{ij}$ = normal time duration for completing job (i, j)
$\underline{\tau}_{ij}$ = the minimum, or crash time duration for completing job (i, j)
α_{ij} = shortening cost in \$/unit time

$\alpha_{ij} \geqq 0$ is the cost for reducing the time required for completing job (i, j) below the normal time $\overline{\tau}_{ij}$, per unit. See Figure 7.6. For all dummy jobs (i, j) we always have $\overline{\tau}_{ij} = \underline{\tau}_{ij} = 0$ and $\alpha_{ij} = 0$.

The time allowed for completing job (i, j), t_{ij}, is itself a variable in this problem, subject to the bounds $\underline{\tau}_{ij} \leqq t_{ij} \leqq \overline{\tau}_{ij}$, and the shortening cost associated with t_{ij} is $(\overline{\tau}_{ij} - t_{ij})\alpha_{ij}$. $\overline{\tau}_{ij} = \underline{\tau}_{ij}$ implies that job (i, j) cannot be shortened, in this case $t_{ij} = \overline{\tau}_{ij} = \underline{\tau}_{ij}$, and we take $\alpha_{ij} = 0$. If $\overline{\tau}_{ij} > \underline{\tau}_{ij}$ and $\alpha_{ij} = 0$ we will obviously select $t_{ij} = \underline{\tau}_{ij}$ as this is likely to reduce project duration at no additional cost. So, we assume that if $\overline{\tau}_{ij} > \underline{\tau}_{ij}$ then $\alpha_{ij} > 0$.

Let t_i be the clock time at which the event corresponding to node i occurs. The t_i, t_{ij} variables have to satisfy $t_j - t_i \geqq t_{ij}$ for each $(i, j) \in \mathcal{A}$ in this problem. Given the job durations, t_{ij}s, the total shortening cost is $\sum((\overline{\tau}_{ij} - t_{ij})\alpha_{ij}$: over $(i, j) \in \mathcal{A})$, and since $\sum(\overline{\tau}_{ij}\alpha_{ij}$: over $(i, j) \in \mathcal{A})$ is a known constant, minimizing the total shortening cost is equivalent to maximizing $\sum(\alpha_{ij}t_{ij}$: over $(i, j) \in \mathcal{A})$. Hence, if λ is the specified project duration, the problem of meeting this deadline with minimum shortening cost is equivalent to : find $(t_{ij} : (i, j) \in \mathcal{A}), (t_i : i \in \mathcal{N})$ that solve (7.3) to (7.6).

$$\begin{array}{rlcll}
Q(\lambda) = & \text{Maximum } \sum(\alpha_{ij}t_{ij} & : & \text{over } (i,j) \in \mathcal{A}) & \\
\text{Subject to} & t_{ij} - (t_j - t_i) & \leqq & 0, \text{ for all } (i,j) \in \mathcal{A} & (7.3) \\
 & t_{ij} & \leqq & \overline{\tau}_{ij}, \text{ for all } (i,j) \in \mathcal{A} & (7.4) \\
 & -t_{ij} & \leqq & -\underline{\tau}_{ij}, \text{ for all } (i,j) \in \mathcal{A} & (7.5) \\
 & t_n - t_1 & \leqq & \lambda & (7.6)
\end{array}$$

The optimum project shortening cost is $P(\lambda) = \sum(\overline{\tau}_{ij}\alpha_{ij}$: over $(i,j) \in \mathcal{A}) - Q(\lambda)$. Associating the dual variables f_{ij}, g_{ij}, h_{ij} to the constraints in (7.3), (7.4), (7.5), respectively, and the dual variable v to the constraint in (7.6), we see that the dual of the above problem is (7.7). The dual problem (7.7) has the structure of a minimum cost flow problem. From the complementary slackness optimality conditions for this primal, dual pair of LPs, it can be seen that if (7.7) has an optimum solution, then it has an optimum solution in which at least one of the two variables g_{ij} or h_{ij} is 0 for each arc $(i,j) \in \mathcal{A}$. The constraints in (7.7) imply that in such an optimum dual solution $g_{ij} = (\alpha_{ij} - f_{ij})^+ =$ max. $\{0, \alpha_{ij} - f_{ij}\}$,

$$\begin{aligned} \text{Min. } W(\lambda, v, f, g, h) = \lambda v &\;+\; \sum_{(i,j)\in\mathcal{A}} (\overline{\tau}_{ij} g_{ij}) - \sum_{(i,j)\in\mathcal{A}} (\underline{\tau}_{ij} h_{ij}) \\ \text{Subject to } f_{ij} + g_{ij} - h_{ij} &\;=\; \alpha_{ij}, \text{ for all } (i,j) \in \mathcal{A} \\ f(i,\mathcal{N}) - f(\mathcal{N},i) &\;=\; \begin{cases} 0, & \text{for all } i \neq 1 \text{ or } n \\ -v, & \text{for } i = n \end{cases} \qquad (7.7) \\ f, g, h, v &\;\geqq\; 0 \end{aligned}$$

and $h_{ij} = (\alpha_{ij} - f_{ij})^- = |$ min. $\{0, \alpha_{ij} - f_{ij}\}|$. So, in such a solution we have $W(\lambda, v, f, g, h) = \lambda v + \sum(\omega_{ij}(f_{ij})$: over $(i,j) \in \mathcal{A})$, where $\omega_{ij}(f_{ij})$ is a piecewise linear convex function defined below (it is convex because $\overline{\tau}_{ij} \geqq \underline{\tau}_{ij}$). See Figure 7.7.

$$\omega_{ij}(f_{ij}) = \begin{cases} -\overline{\tau}_{ij}(f_{ij} - \alpha_{ij}) \text{ if } f_{ij} \leqq \alpha_{ij} \\ -\underline{\tau}_{ij}(f_{ij} - \alpha_{ij}) \text{ if } f_{ij} > \alpha_{ij} \end{cases}$$

By these arguments, the dual (7.7) is equivalent to the minimum cost flow problem (7. 8) in the network G, with a piecewise linear convex objective function. We can of course solve the original project shortening cost problem (7.3) to (7.6) as an LP, but as its dual is a minimum cost flow problem for which there are very efficient special algorithms, it turns out to be much more convenient to solve the dual problem using these algorithms, and then obtain an optimum solution

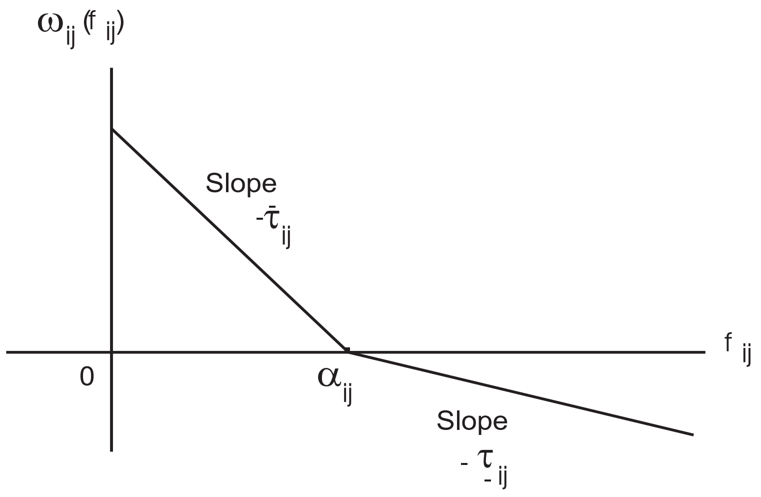

Figure 7.7:

of the primal problem by the complementary slackness conditions for optimality.

If $(i,j) \in \mathcal{A}$ is such that $\overline{\tau}_{ij} = \underline{\tau}_{ij}$, then $t_{ij} = \overline{\tau}_{ij} = \underline{\tau}_{ij}$ is known and fixed, in this case $\omega_{ij}(f_{ij})$ defined above is in fact linear. $\omega_{ij}(f_{ij})$ is piecewise linear convex, and not linear, only if $\overline{\tau}_{ij} > \underline{\tau}_{ij}$. (7.8) can be transformed into a linear minimum

$$\begin{array}{rcl} \text{Minimize } \lambda v + \sum(\omega_{ij}(f_{ij}) & : & \text{over } (i,j) \in \mathcal{A}) \\ \text{Subject to } f(i,\mathcal{N}) - f(\mathcal{N},i) & = & \left\{ \begin{array}{l} \;\;\,0, \text{ for all } i \neq 1 \text{ or } n \\ -v, \text{ for } i = n \end{array} \right. \\ f, v & \geqq & 0 \end{array} \qquad (7.8)$$

cost flow problem as in Section 5.9. For this we replace each arc (i,j) on which ω_{ij} is not linear, by the pair of arcs $(i,j)_1, (i,j)_2$ called Type 1 and Type 2 arcs, with the following data (see Figure 7.8):

lower bounds on all the arcs $= 0$
capacity of $(i,j)_1$ is $k(i,j,1) = \infty$
capacity of $(i,j)_2$ is $k(i,j,2) = \alpha_{ij}$

unit cost coefficient on $(i,j)_1 = c(i,j,1) = -\underline{\tau}_{ij}$
unit cost coefficient on $(i,j)_2 = c(i,j,2) = -\overline{\tau}_{ij}$.

If arc (i,j) is such that $\overline{\tau}_{ij} = \underline{\tau}_{ij}$, it is treated as a Type 1 arc itself, and no Type 2 arc corresponding to it is introduced (since $\omega_{ij}(f_{ij})$ is linear for such arcs). See Figure 7.8.

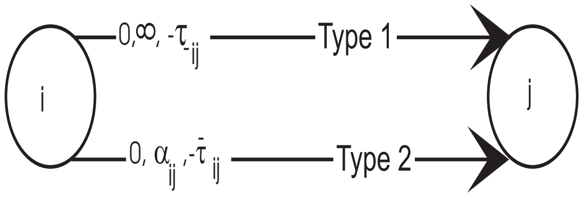

Figure 7.8: If $\bar{\tau}_{ij} > \underline{\tau}_{ij}$, arc (i,j) corresponds to the pair of Type 1, 2 arcs with this data (lower bound, capacity, unit cost) in $\mathcal{A}'$. If $\bar{\tau}_{ij} = \underline{\tau}_{ij}$, there will be no Type 2 arc.

Let $\mathrm{G}' = (\mathcal{N}, \mathcal{A}')$ be the augmented network with the data on arcs as defined above. Let $f(i,j,r)$ denote the flow amount on the Type r arc $(i,j)_r \in \mathcal{A}'$ for $r = 1, 2$, and let $f' = (f(i,j,r) : (i,j,r) \in \mathcal{A}')$. Given a feasible flow vector $f' = (f(i,j,r))$ in G$'$, the corresponding flow vector $f = (f_{ij})$ feasible to (7.8) in the original network G is obtained from

$$f_{ij} = \begin{cases} f(i,j,1) + f(i,j,2) & \text{if both Type 1,2 arcs corresponding to} \\ & (i,j) \text{ exist in G}' \\ f(i,j,1) & \text{otherwise} \end{cases}$$

(7.8) is equivalent to the problem of minimizing $\lambda v + \sum(c(i,j,r)f(i,j,r) :$ over $(i,j)_r \in \mathcal{A}', r = 1,2)$ in G$'$. The flow vector f in G corresponding to an optimum flow vector f' in G$'$, is an optimum flow vector for (7.8).

Let $\overline{\lambda}, \underline{\lambda}$ be the lengths of the critical paths in G with $(\bar{\tau}_{ij})$, $(\underline{\tau}_{ij})$ as the arc length vectors respectively. When $\lambda = \overline{\lambda}$, the optimum $t_{ij} = \bar{\tau}_{ij}$ for each $(i,j) \in \mathcal{A}$, and the optimum shortening cost is 0. $\underline{\lambda}$ is at the other end, it is the minimum project duration. Since the dual problem is (7.8) is always feasible, the objective function must be unbounded below in it (and the same thing happens in G$'$) whenever $\lambda < \underline{\lambda}$, and vice versa.

We would like to solve the project shortening cost minimization problem parametrically in λ as it decreases from $\bar{\lambda}$ to $\underline{\lambda}$. For this, we need to solve the minimum cost flow problem of minimizing $\lambda v + \sum(c(i,j,r)f(i,j,r)$: over $(i,j)_r \in \mathcal{A}', r = 1,2)$ in G$'$, treating λ as a parameter. This problem is in the same form as the parametric maximum profit flow problem discussed in Section 5.3, with the exception that the objective function here is to be minimized instead of being maximized, so it can be solved by an appropriate modification of that algorithm. The node price vector $\pi = (\pi_i)$ in G$'$ in this algorithm turns out to be the vector of early occurrence times, (t_i), for nodes i in G, associated with the present λ. If π is the node price vector in G$'$, the relative cost coefficients wrt it for Type 1, 2 arcs $(i,j)_1$, $(i,j)_2 \in \mathcal{A}'$ are:

$$\begin{aligned} \bar{c}(i,j,1) &= c(i,j,1) + (\pi_j - \pi_i) = -\underline{\tau}_{ij} + (\pi_j - \pi_i) \\ \bar{c}(i,j,2) &= c(i,j,2) + (\pi_j - \pi_i) = -\overline{\tau}_{ij} + (\pi_j - \pi_i) \end{aligned}$$

The signs of these relative cost coefficients are opposite to those in Section 5.3 since we discussed the maximum profit flow problem there, but we have a minimum cost flow problem in G$'$. The feasible flow vector, node price vector pair $(f' = (f(i,j,r)), \pi = (\pi_i))$ is optimal in G$'$ for a given value of λ if the following optimality conditions hold.

$$\begin{aligned} \bar{c}(i,j,1) &\geqq 0 \text{ for all } (i,j)_1 \in \mathcal{A}' \\ \bar{c}(i,j,1) &> 0 \Rightarrow f(i,j,1) = 0 \\ \text{if } (i,j)_2 \in \mathcal{A}' \quad &, \quad \text{then} \begin{cases} \bar{c}(i,j,2) > 0 \Rightarrow f(i,j,2) = 0 \\ \bar{c}(i,j,2) < 0 \Rightarrow f(i,j,2) = k(i,j,2) \end{cases} \qquad (7.9) \\ \text{and } \pi_n - \pi_1 &= \lambda \end{aligned}$$

Let $(f'^p, \pi^p = (\pi^p))$ be a feasible flow vector, node price vector pair with the value of f'^p being v^p, satisfying (7.9) in G$'$ for $\lambda = \lambda_p$. To find optimum flow vectors in G$'$ for $\lambda < \lambda_p$, the parametric algorithm applies the labeling algorithm to increase the flow value while continuing to satisfy the optimality conditions (7.9) keeping $\lambda = \lambda_p$. Only arcs $(i,j)_r \in \mathcal{A}'$ for which $\bar{c}(i,j,r) = 0$ are **admissible** for flow change in this step. Labeling is carried out in two stages in this step. In Stage 1 we check whether the flow value can be increased to ∞. If this is possible, it would imply that if λ is decreased from its present value λ_p, the objective value in G$'$ becomes unbounded below, which in turn implies that the original problem (7.3) to (7.6) becomes infeasible, i.e., $\lambda_p = \underline{\lambda}$, the minimum possible project duration under crashing. Another explanation for this is the following. The flow value can be increased to ∞ making flow changes on admissible arcs only, iff there exists a chain from node 1 to node n, say $\mathcal{C}$, consisting of admissible Type 1 arcs only, since only they have infinite capacity. Hence $\bar{c}(i,j,1) = -\underline{\tau}_{ij} + \pi^p_j - \pi^p_i = 0$, or $\pi^p_j - \pi^p_i = \underline{\tau}_{ij}$, for each (i,j) on $\mathcal{C}$. Hence $\mathcal{C}$ is a critical path, and each job on it on it is at its crash time, which implies that $\pi^p_n = \lambda_p = \underline{\lambda}$, and hence λ cannot be reduced below its current value.

In Stage 1, if it has been verified that the flow value cannot be increased to ∞, in Stage 2 we obtain the maximum flow value making flow changes on admissible arcs of both Types 1 and 2.

During the labeling process, if an FAP from node 1 to node n consisting of admissible Type 1 arcs only has been identified (which only happens if n is labeled in Stage 1) we say that an **infinite breakthrough** has occurred, this is a signal that the current value of λ is $\underline{\lambda}$. Any FAP identified during Stage 2 will consist of some Type 2 arcs which have finite capacities, and we refer to its occurrence as a **finite breakthrough**.

The label on a node j in this algorithm will be of the form $(i, \pm, r = 1 \text{ or } 2)$. If it is $(i, +, r)$, it means that $(i,j)_r$ is a forward arc on the FAP from 1 to j. If it is $(i, -, r)$ it means that $(j,i)_r$ is a reverse arc on the same FAP.

THE PARAMETRIC SHORTENING COST MINIMIZATION ALGORITHM

Initial Step Find the longest chains from node 1 to all the other nodes in G, using $\overline{\tau}_{ij}$s, normal durations, as the lengths of arcs $(i,j) \in \mathcal{A}$, by the forward pass routine discussed earlier. For $i \in \mathcal{N}$ let (L_i, t^1_i) be the forward pass label on i. Define $\pi^1 =$

(π_i^1), where $\pi_i^1 = t_i^1$ for each $i \in \mathcal{N}$. These node prices satisfy $\pi_j^1 - \pi_i^1 \geqq \bar{\tau}_{ij}$, which is opposite to those in Chapter 4, since we are finding the longest chains here. Define $f'^1 = 0$. It can be verified that (f'^1, π^1) satisfies (7.9) when $\lambda = \bar{\lambda} = \lambda_1$, and hence is an optimum pair in G$'$ for λ_1. When $\lambda = \lambda_1$, $t_{ij} = \bar{\tau}_{ij}$ for each $(i, j) \in \mathcal{A}$, and $t_i = \pi_i^1$ is the earliest occurrence time for the event associated with node i, $i \in \mathcal{N}$. Enter the labeling routine.

Stage 1 of the Labeling Routine

Labeling Step 1 Label the start node 1 with $(\emptyset, +)$. List = { 1 }.

Labeling Step 2 Select a node for Stage 1 scanning Let λ_p be the current value of λ, and (f'^p, π^p) the present optimum pair with v^p as the value of f'^p. If list = $\emptyset$, go to Stage 2 of the labeling routine. Otherwise select the node from the top of the list to scan, delete it from the list, and go to Labeling Step 3.

Labeling Step 3 Stage 1 scanning Let i be the node to be scanned. Label all unlabeled nodes j such that $(i, j, 1)$ is admissible for flow change, with $(i, +, 1)$, and include them at the bottom of the list as they are labeled.

If node n is now labeled, there is an infinite breakthrough. This implies that $\lambda_p = \underline{\lambda}$, terminate the algorithm. If n is not yet labeled, go back to Labeling Step 2.

Stage 2 of the Labeling Routine Make the list = set of all labeled nodes at this time

Labeling Step 4 Select a node for Stage 2 scanning If list = $\emptyset$, go to the node price change step. Otherwise select the node from the top of the list to scan, delete it from the list, and go to Labeling Step 5.

Labeling Step 5 Stage 2 scanning Let i be the node to scan.

Forward labeling Identify all unlabeled nodes j such that (i, j, r) is admissible for flow change, and $f^p(i, j, r) < k(i, j, r)$ for $r = 1$ or 2 or both, then label j with $(i, +, r)$ with any of the r satisfying the above condition and include j at the bottom of the list as it is labeled.

Reverse labeling Identify all unlabeled nodes j such that (j, i, r) is admissible for flow change, and $f^p(j, i, r) > 0$ for $r = 1$ or 2 or both, then label j with $(i, -, r)$ with any of the r satisfying the above condition, and include j at the bottom of the list as it is labeled.

If node n is now labeled, there is a finite breakthrough, go to the flow augmentation step. Otherwise go back to Labeling Step 4.

Flow augmentation Trace the admissible FAP from node 1 to node n using the node labels, and carry out flow augmentation using it. Erase the labels on all the nodes and go back to Labeling Step 1.

Node price change Let $\mathbf{X}$, $\overline{\mathbf{X}}$ be the current sets of labeled, unlabeled nodes respectively. Define

$$\begin{aligned}
\mathbf{A}^1 &= \{(i,j)_r : r = 1 \text{ or } 2, (i,j)_r \in (\mathbf{X}, \overline{\mathbf{X}}), \\
&\qquad \text{and current } \bar{c}(i,j,r) > 0\} \\
\mathbf{A}^2 &= \{(i,j)_r : r = 1 \text{ or } 2, (i,j)_r \in (\overline{\mathbf{X}}, \mathbf{X}), \\
&\qquad \text{and current } \bar{c}(i,j,r) < 0\} \\
\delta_1 &= \text{min. } \{\bar{c}_(i,j,r) : (i,j)_r \in \mathbf{A}^1\} \\
\delta_2 &= \text{min. } \{-\bar{c}_(i,j,r) : (i,j)_r \in \mathbf{A}^2\} \\
\delta &= \text{min. } \{\delta_1, \delta_2\} \\
\pi_i^p(\nu) &= \begin{cases} \pi_i^p, & \text{if } i \in \mathbf{X} \\ \pi_i^p - \nu, & \text{if } i \in \bar{\mathbf{X}} \end{cases}
\end{aligned}$$

for $0 \leqq \nu \leqq \delta$. Let $\pi^p(\nu) = (\pi_i^p(\nu) : i \in \mathcal{N})$. It can be verified that the feasible pair $(f'^p, \pi^p(\nu))$ satisfies the optimality conditions (7.9) when $\lambda = \pi_n^p - \nu$ for all $0 \leqq \nu \leqq \delta$. So, if we define

$$\begin{aligned} t^p_{ij}(\nu) &= Min.\{\bar{\tau}_{ij}, \pi^p_j(\nu) - \pi^p_i(\nu)\}, \text{ for } (i,j) \in \mathcal{A} \\ t^p_i(\nu) &= \pi^p_i(\nu), \text{ for } i \in \mathcal{N} \end{aligned}$$

then $(t^p_{ij}(\nu) : (i,j) \in \mathcal{A}), (t^p_i(\nu) : i \in \mathcal{N})$ is an optimum solution for the original project shortening cost minimization problem (7.7) when $\lambda = \lambda_p - \nu = \pi^p_n - \nu$, for $0 \leqq \nu \leqq \delta$. Since there is no change in the flow vector, the optimum objective value in G′ (which is equal to $Q(\lambda)$ by the duality theorem of LP) decreases with slope v^p as λ decreases in this interval from λ_p to $\lambda_{p+1} = \lambda_p - \delta$. Hence, the minimum job shortening cost $P(\lambda)$ increases with slope v^p as λ decreases in this interval. In other words, $P(\lambda)$ is linear in this interval with slope $-v^p$.

Define $\pi^{p+1} = \pi^p(\delta), \lambda_{p+1} = \lambda_p - \delta$. Take (f'^p, π^{p+1}) as the new pair in G′, λ_{p+1} as the new value for λ, include all the labeled nodes in the list, and resume labeling by going back to Labeling Step 2 in Stage 1.

Discussion

In the set $\mathbf{A}^2$ defined in a node price change step, all arcs are always saturated Type 2 arcs. Likewise, all arcs in the set $\mathbf{A}^1$ have zero flow.

The set $\mathbf{A}^1$ is always nonempty in a node price change step. The reason for this is the following. Let node q be the unlabeled node with the smallest number in G′. The only node in G′ which has no arcs incident into it is 1 and it is labeled. So, there exists an arc of the form $(i,q) \in \mathcal{A}'$. By the acyclic numbering of the nodes in G, $i < q$, and from the definition of q, i must be in $\mathbf{X}$. If (i,q) is currently admissible, node q would have been labeled when node i is scanned, a contradiction. So, (i,q) is inadmissible, and hence by (7.9), the current value of $\bar{c}(i,q,1) > 0$. So, $(i,q)_1 \in \mathbf{A}^1$. Thus $\mathbf{A}^1 \neq \emptyset$. Hence the quantity δ in a node price change step in this algorithm is always positive and finite.

The algorithm obtains the optimum job durations, and the earliest occurrence times for the events associated with the nodes in the

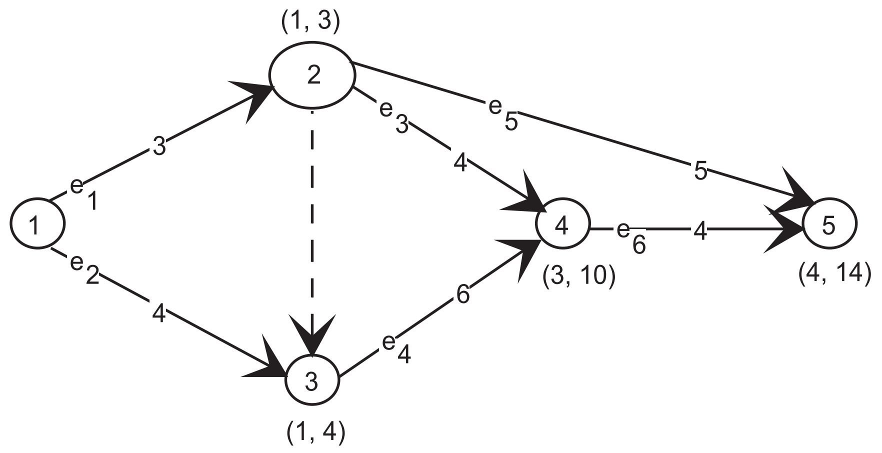

Figure 7.9:

network, $(t_{ij}), (t_i)$, corresponding to each value of λ in its range. For any λ, the latest occurrence times associated with the nodes, can be obtained using $t_n = \lambda$ and the job duration values (t_{ij}) for that λ, by applying the backward pass routine. These provide all the necessary information to the scheduler to identify all the critical jobs, to compute the total slack of each job, and to schedule the jobs over time for that value of λ.

As mentioned above, the optimum job shortening cost, $P(\lambda)$, increases as λ decreases. Whenever flow augmentation occurs, the slope of $P(\lambda)$ as λ decreases, increases. We have already seen that $P(\lambda)$ is piecewise linear. Hence, $P(\lambda)$ is a piecewise linear convex function.

As an example consider the project consisting of six jobs denoted by e_1 to e_6, with data given in the following table. The arrow diagram for the project is given in Figure 7.9. Numbers on the arcs there are the normal durations, and the entries by the side of the nodes are the forward pass labels corresponding to these normal durations.

In Figure 7.10 we show the augmented network G$'$. The entries on arc (i, j, r) in this figure are the capacity $k(i, j, r)$ and the cost coefficient $c(i, j, r)$. The Type 1 arcs are all the arcs with ∞ capacity. All lower bounds are 0. The initial node prices in π^1, from the earliest

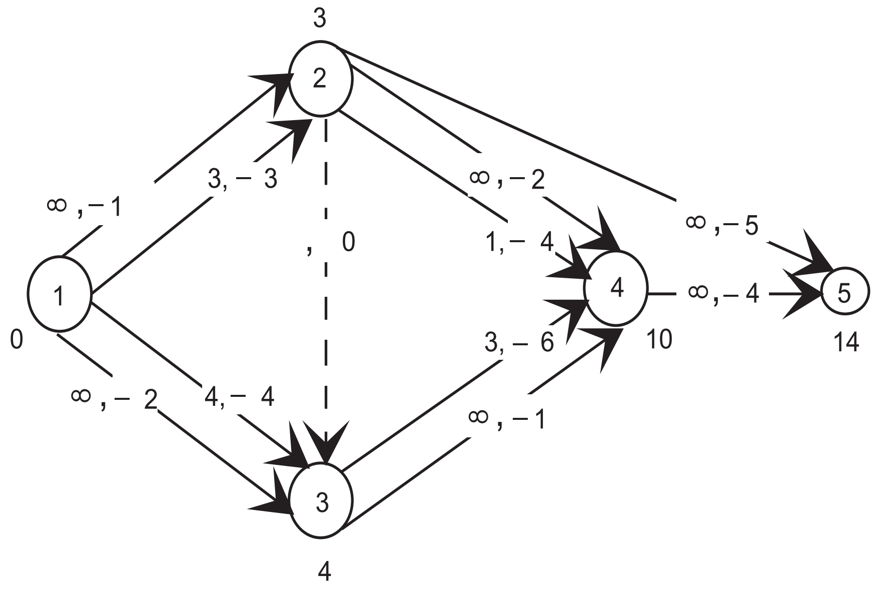

Figure 7.10: All lower bounds are zero. Data on the arcs is capacity, unit cost.

Job	Immediate predecessors	Normal duration	Crash duration	Unit shortening cost
e_1		3	1	3
e_2		4	2	4
e_3	e_1	4	2	1
e_4	e_1, e_2	6	1	3
e_5	e_1	5	5	0
e_6	e_3, e_4	4	4	0

occurrence times associated with the nodes, under normal job durations, are entered by the side of the nodes. The arcs admissible for flow change in Figure 7.10 are $(4, 5)_1, (3, 4)_2, (1, 2)_2$ and $(1, 3)_2$. After labeling and flow augmentation, we get the flow vector of value $v^1 = 3$ (this flow vector is marked in Figure 7.11 with nonzero flow amounts entered inside little boxes by the side of the arcs) and reach the node price change step with the cut $(\mathbf{X}, \overline{\mathbf{X}}) = (\{1, 2, 3\}, \{4, 5\})$. So, $\mathbf{A}^1 =$

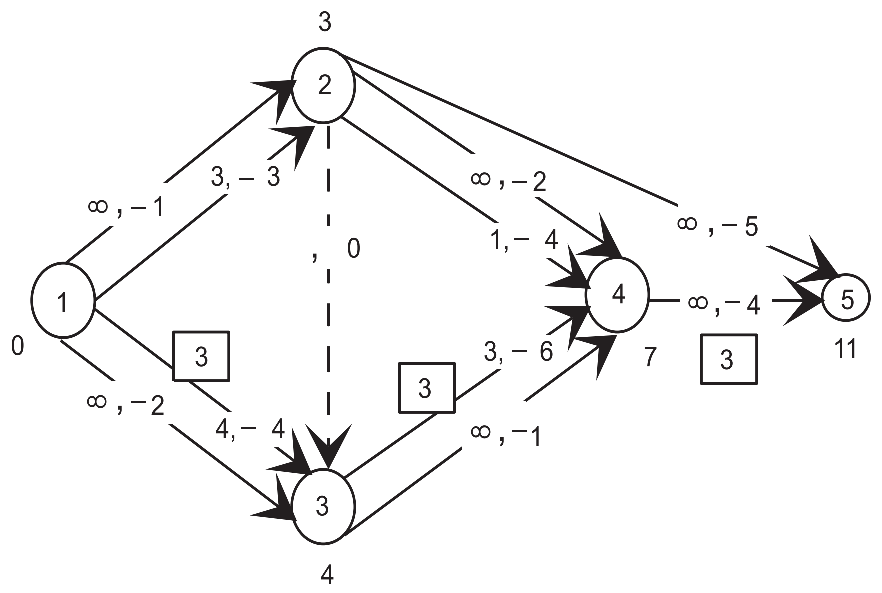

Figure 7.11: Data on arcs is capacity, unit cost. Nonzero flow amounts entered in little boxes by side of the nodes. Present node prices are entered by the side of the nodes.

$\{(2,4)_2,(2,4)_1,(3,4)_1,(2,5)_1,\}$, $\mathbf{A}^2 = \emptyset$, and $\delta =$ min. $\{3,\ 5,\ 5,\ 6\} = 3$. Hence for project duration $\pi_5^1 - \nu = 14 - \nu$, the earliest occurrence times associated with the nodes 1 to 5 in that order are (0, 3, 4, $10-\nu$, $14-\nu$), for $0 \leqq \nu \leqq 3$. So, when the project duration is $14-\nu$, the optimum job durations for e_1 to e_6 in that order are (3, 4, 4, $6-\nu$, 5, 4), for $0 \leqq \nu \leqq 3$, and the optimum shortening cost increases with slope $v^1 = 3$ as project duration decreases from 14 to 11. Making $\nu = \delta = 3$, we get the optimum flow vector, node price vector pair marked in Figure 7.11.

The algorithm can be continued in the same manner. It terminates with an infinite breakthrough when the project duration reaches 7. The project shortening cost curve for this project is shown in Figure 7.12.

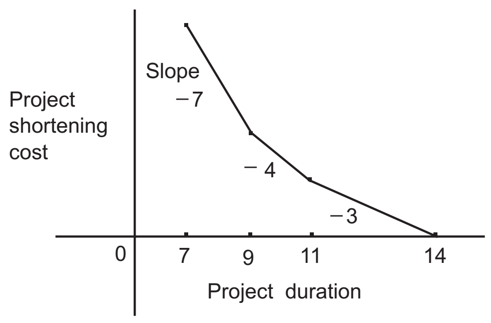

Figure 7.12:

7.3 Resource Constrained Project Scheduling Problems

In the CPM models discussed so far, we assumed that the only constraints in scheduling jobs over time are those imposed by the predecessor relationships among the jobs. 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**. See Exercise 7.6, for a problem of this type. Usually, solving resource constrained project scheduling cannot be accomplished purely by network techniques alone; typically, they require combinatorial optimization methods. Also, practical resource constrained project scheduling normally leads to very large combinatorial optimization problems, for which efficient exact algorithms are not available at the moment. Hence, a variety of heuristic algorithms have been developed for resource constrained project scheduling, see Battersby [1967], Burman [1972], Elmaghraby [1977], Weist and Levy [1977], and Willis and Hastings [1976].

7.4 PERT

In the CPM models discussed so far, the job durations are assumed to be either known constants, or deterministic variables whose values can be selected from known intervals by spending a deterministic sum of money. In real world project scheduling, job durations may not be known with certainty, in fact there may be quite a bit of uncertainty or random variation in them. Uncertainty in job durations appears most often in research and development projects, in projects dealing with designing or launching a new product, etc. The PERT (Program Evaluation and Review Technique) model deals with project scheduling under such uncertainty.

The PERT model usually assumes that the various job durations are mutually independent random variables. Replacing each of these random variables by their expected values leads to a deterministic problem which can be analyzed using the CPM models already discussed. This often leads to an optimistic estimate of the expected project duration. In engineering problems, the expected value of a job duration can itself be approximated by a convex combination of the most probable job duration, an optimistic job duration, and a pessimistic job duration, all guessed by qualified project engineers; PERT normally uses this approach to estimate the expected job durations.

If the job durations can be assumed to be random variables with known probability distributions, then a simulation can be run by selecting values for job durations from these distributions. Once the job durations are known, the critical path and the scheduling information can be obtained using the CPM methods. The procedure can be repeated many times by selecting different sets of values for the random variables. From these simulation runs, information like the average project duration and its standard deviation, probabilities for the various jobs being critical, etc., can be computed. Statistical analysis of the data from these simulation runs can give the scheduler very useful information. Because of the random nature of job durations, it is not possible to lay down a rigid time schedule for the jobs at the beginning of the project, and expect to stick with it. A rough time schedule is prepared using the information from CPM based on expected job durations, and the simulation runs. As the event corresponding to each node in the arrow diagram materializes, a review is made, and the time schedule for the remaining jobs is revised. See Burman [1972], Elmaghraby [1977], Weist and Levy [1977].

7.5 Exercises

7.4 Let G $= (\mathcal{N}, \mathcal{A})$ be the arrow diagram for a project. λ is the project completion time, and d_1 denotes a target value for λ. For each job p, $\underline{\tau}_p, \bar{\tau}_p, \alpha_p$ have the same meaning as in Section 7.2. For each unit of time the project is completed before the target time d_1, there is a profit of δ\$. If $\lambda > d_1$, a penalty is incurred, this penalty, denoted by $f(\lambda)$, is 0 if $\lambda \leqq d_1$, and a positive piecewise linear function of λ in the range $\lambda > d_1$, with slopes given below.

Interval	Slope of $f(\lambda)$
$d_1 < \lambda \leqq d_2$	g_1
$d_2 < \lambda \leqq d_3$	g_2
$\vdots$	$\vdots$
$d_u < \lambda$	g_u

where $d_1 < d_2 < \ldots < d_u$ and $g_1 < g_2 < \ldots < g_u$. Define the net cost of the project to be the cost of job shortening + $f(\lambda)$ − any profit due to project completion before target completion date d_1. Formulate the problem of finding an optimum project duration to minimize the net cost as an LP, and show how it can be solved using a network flow approach. As a numerical example consider the following project.

The plant is scheduled for erection and commissioning in $d_1 = 36$ months from land acquisition. There is a profit of \$35 million/month if the plant is completed before 36 months. If the erection division fails to hand over the plant to the customer at the end of 36 months, there

Project: Setting Up a Fossil Fuel Power Plant

No. p	Job	IPs	$\overline{\tau}_p$	$\underline{\tau}_p$ if $< \overline{\tau}_p$	α_p
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	1	6
10.	Control room civil works	4	5	2	3
11.	TG civil works	5	9	7	15
12.	Training	8	6		
13.	Boiler final civil works	9	9	8	15
14.	Erection of control room	10	8	7	5
15.	Erection of TG	6, 11	10	8	20
16.	Boiler erection	7, 13	12	11	35
17.	Hydraulic test	16	2		
18.	Boiler light up	14, 17	1.5	0.5	7.5
19.	Box up of turbine	15	3	2	15
20.	Steam blowing, safety valve floating	18, 19	2.5	1.5	10
21.	Turbine rolling	20	1.5	1	20
22.	Trial run	21	1	0.5	25
23.	Synchronization	22	1	0.5	20

IP = Immediate predecessors, $\overline{\tau}_p, \underline{\tau}_p$ in months, α_p in \$mil.

will be a penalty with increasing slopes of \$30, 35, 40, and 55 million beyond 36, 37, 38, and 39 months, respectively. Solve this problem and obtain an optimum project schedule. (Kanda and Singh[1988])

7.5 How does the formulation in Exercise 7.4 change if the profit for early completion is a constant, \$ ξ, irrespective of what the value of λ is $< d_1$, but everything else remains unchanged.

7.6 Coke Depot Project A depot is to be built to store coke and

Job	IPs	Duration	No. of resources needed each week of job					
			R_1	R_2	R_3	R_4	R_5	R_6
1. B. piling		5	2	1				
2. Clear site for SH.		8	6					1
3. B. excavation for cols.	1	4	4					1
4. SH. excavations for C.	2, 3	4	4					1
5. Concrete tops of piles for B.	3	3	2		2			
6. Place cols. for B.	5	4	3	1			1	
7. Excavate access road	5	4	3					1
8. Put in B.	6	3		2				
9. Stairways inside B.	6	1		2				
10. Excavate pit for WB.	4	6	1					1
11. Concrete for SH.	4	12	2		4			
12. Main C. foundation	4	4	1					
13. Brick walls for B.	8, 9	3	2			2		
14. Clad in steel for B.	8, 9	1		2				
15. Install internal equip. in B.	8, 9	6	2	1		2		
16. Erect gantry for main C.	12, 6	1	2	1			1	
17. Install C. under hoppers	11	1	2	2				
18. Concrete pit for WB.	11, 10	2	1		2			
19. Excavate for hard-standing	7	9	4					1
20. Lay access roadway	7	9	4					
21. Install outloading equip. for B.	15	2	2	3				
22. Line B.	13, 14	1	1	1				
23. Install main C.	16	1	2	2				
24. Build weighhouse	18	4	1			2		
25. Erect perimeter fence	19	4	2					
26. Install C. to SH.	17, 23	1	2	2				
27. Install WB.	24	1	2	2			1	
28. Lay hard-standing	19, 18	6	4					
29. Commission hoppers	26	1						

IP = Immediate predecessors

to load and dispatch trucks. There will be three storage hoppers (SH. in abbreviation), a block of bunkers (B. in abbreviation), interconnecting conveyors (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, the duration (in weeks) and resource requirements of each job are given in the table above.

There are six resources required for construction, their availability is limited to the quantities given below during the construction. 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 without the resource availabilities being exceeded using an appropriate heuristic approach. (Willis and Hastings[1976])

Resource	Symbol	Available quantity
Laborers	R_1	10
Steel men	R_2	5
Concrete men	R_3	4
Bricklayers	R_4	2
Cranes	R_5	1
Dumpers	R_6	2

7.7 Draw an arrow diagram for each of the following projects. For the values of project duration in its feasible range, obtain the optimum job durations and the earliest occurrence times of events associated with nodes in the network, and draw the project shortening cost curve as a function of the project completion time in each case. (R. Visweswara Rao).

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

No. p	Activity	IPs	Duration (days)	α_p
1.	Prel. Syst. description		40	
2.	Develop specs.	1	100	
3.	Client approval & place order	2	50	
4.	Develop I/O summary	2	40 - 60	15
5.	Develop alarm list	4	40	
6.	Develop log formats	3, 5	40	
7.	Software def.	3	35	
8.	Hardware requirements	3	35	
9.	Finalize I/O summary	5, 6	50 - 60	18
10.	Anal. performance calculation	9	50 - 70	20

Contd.

(a) contd.

No. p	Activity	IPs	Duration (days)	α_p
11.	Auto. turbine startup anal.	9	60	
12.	Boiler guides anal.	9	30	
13.	Fabricate & ship	10, 11, 12	400	
14.	Software preparation	7, 10, 11	60 - 80	22
15.	Install & check	13, 14	100 - 130	30
16.	Termination & wiring lists	9	30	
17.	Schematic wiring lists	16	60	
18.	Pulling & term. of cables	15, 17	60	
19.	Operational test	18	80 - 125	30
20.	First firing	19	1	

IP = Immediate predecessors

α_p = shortening cost of job p/day shortened

(b) Electrical Auxiliary System Design for a Nuclear Plant

No. p	Activity	IPs	Duration (days)	α_p
1.	Aux. load list		100 - 120	15
2.	13.8 switchgear load ident.	1	140 - 190	12
3.	4.16kv & 480 v. switchgear load ident.	1	45	
4.	Vital AC load determination	1	200 - 300	14
5.	DC load determ.	1	110 - 165	18
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 current study	6, 7	20	

Contd.

(b) Contd.

No. p	Activity	IPs	Duration (days)	α_p
12.	Power transformer sizing	2, 11	80	
13.	Composite oneline diagram	2,3	72	
14.	Safety (class 1E) system design	13	150 - 200	25
15.	Non-class 1E system design	13	160 - 190	20
16.	Relaying oneline & metering dia.	13	80	
17.	3-line diagram	14, 15, 16	150	
18.	Synchronizing & phasing diagrams	17	100	
19.	Client review	10, 18	25	
20.	Equipment purchase & installation	19	800	

(c) Sewer and Waste System Design for a Power Plant

No. p	Activity	IPs	Duration (days)	α_p
1.	Collection system outline		25 - 40	10
2..	Final design & approval	1	30	
3.	Issue construction drawings	2	23 - 30	8
4.	Get sewer pipe & manholes	1	145	
5.	Fabricate & ship	3,4	45	
6.	Treat. system drawings & approval		50 - 70	15
7.	Issue treat. system construction drawings	6	30	
8.	Award contract	7	60	
9.	Final construction	8, 5	200 - 300	30

IP = Immediate predecessors
α_p = shortening cost of job p/day shortened

7.8 In the job crashing model discussed in Section 7. 2, intuitively it seems correct to assume that if a job is crashed in an optimum schedule for a project duration, then that job will stay crashed in optimum schedules as the project duration decreases further. Show that this could be wrong, using the following example.

Job no. r	IPs	Duration		α_r
		$\underline{\tau}_r$	$\overline{\tau}_r$	
1.		1	3	3
2.		2	4	1
3.	1	0	2	1
4.	1	2	5	1
5.	2, 3	1	6	3

IP = Immediate predecessors

α_r = shortening cost of job r/ unit time shortened

(Ford and Fulkerson [1962 of Chapter 1])

7.9 Consider the project with precedence relationships given below. Draw the arrow diagram for it using the smallest number of nodes. Let G be this arrow diagram. Show that it is possible to decrease the number of dummy arcs in G by increasing the number of nodes. Using this example show that it may not be possible to minimize the number of arcs and the number of nodes in the arrow diagram for a project simultaneously even if there are no parallel activities in it.

Jobs	Immediate Predecessors
a, b, b', c, g, p, l	None
e	a, b, b', p
f	b, b', c, l
d	g, b, b', c
h	a
i	b
j	c
m	b'

(Syslo[1984])

7.10 Develop an efficient algorithm to check whether a given project can be represented by an arrow diagram using no dummy arcs at all. (Syslo[1984])

7.11 Let H = ($\mathbf{N}$, $\mathbf{A}$) be a graph with $\mathbf{N} = \{ 1, \ldots, n \}$, $\mathbf{A} = \{ e_1, \ldots, e_m \}$, in which each node has degree 2 or 3. A *node cover* for H is a subset of nodes in $\mathbf{N}$ which covers all the edges in $\mathbf{A}$.

Define a project with $n+m+1$ activities numbered 1 to $n+m+1$, related to the nodes and edges in H with precedence relations among them defined by the data in H as follows: for $i = 1$ to n activity i in the project corresponds to node i in H; for $p = 1$ to m activity $n+p$ in the project corresponds to edge e_p in H; activity $n+m+1$ is an additional activity; for $p = 1$ to m and $i = 1$ to n, activity $n+p$ is a predecessor of activity i if e_p is incident to i in H; and for all $p = 1$ to m, activity $n+p$ is a predecessor of activity $n+m+1$.

In drawing the arrow diagram for this project, since each of the activities $n{+}1$ to $n{+}m$ have no predecessors, they can be represented by arcs with the same tail node (this is the *start* node). Similarly activities $1, \ldots, n, n+m+1$ have no successors, so they can be represented by arcs with the same head node (this is the *finish* node) in the arrow diagram. When drawn in this way, show that the minimum number of dummy activities needed to represent this project is precisely the minimum number of nodes that cover all the edges in H. Since the problem of finding a minimum cardinality node cover in H is known to be NP-hard, this establishes that the problem of drawing an arrow diagram for a project using the smallest number of dummy arcs is also NP-hard. (Krishnamoorthy and Deo [1979])

7.12 Let G = $(\mathcal{N}, \mathcal{A})$ be the arrow diagram for a project. Consider the project shortening cost minimization problem on G. In addition to the features discussed in Section 7.2, suppose a subset of nodes $\mathbf{P} \subset \mathcal{N}$ called *penalty nodes* is specified, with a due date of d_i for $i \in \mathbf{P}$. Nodes in $\mathbf{P}$ correspond to *key events*, and for each i in it, a penalty of p_i\$ is levied per unit time delay of event associated with it beyond its due date d_i. It is required to minimize the total cost of shortening the activities and the penalties of violating the target dates of key events, treating the project duration λ as a parameter. Develop a modification of the algorithm discussed in Section 7.2 to solve this problem. Apply this algorithm on the project network given in Figure 7.13. (Kanda and Rao [1984])

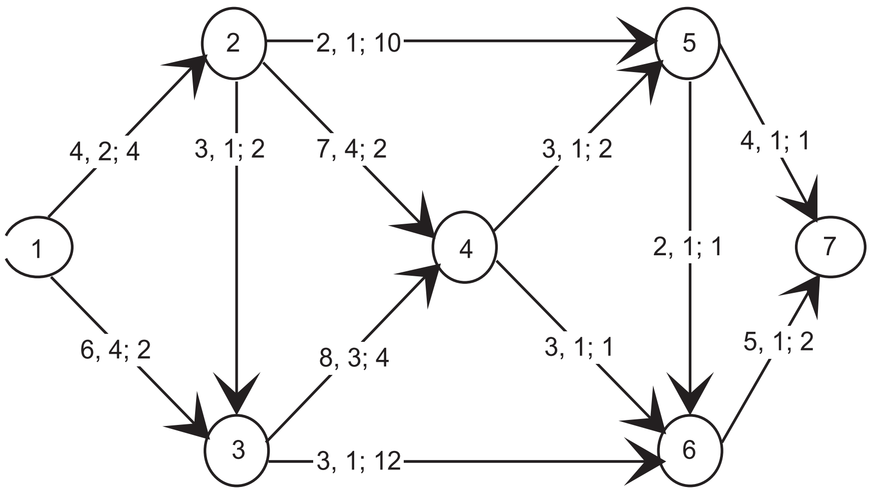

Figure 7.13: Data on arc (i, j) is $\bar{\tau}_{ij}, \underline{\tau}_{ij}.\alpha_{ij}$, in that order. Penalty nodes are 4 ($d_4 = 12$, $p_4 = 10$) and 7 ($d_7 = 18$, $p_7 = 20$).

7.13 Let G = $(\mathcal{N}, \mathcal{A})$ be the AON network for a project. Each job takes exactly one day to process. On each day, any number of jobs can be processed provided they are all unrelated and each of their predecessors has all been processed already, i.e., if job i is processed on day t and $(i, j) \in \mathcal{A}$, then job j can be processed on the $t + 1$th day or any later day. c_{it} is the cost of doing job i on day t. All c_{it} are given and they are all > 0.

Define decision variables $x_{it} = 1$ if job i is processed on the tth day or before, 0 otherwise. Formulate the problem of completing the project at minimum cost using these decision variables. Show how this problem can be solved using the algorithms discussed in Chapter 2. (G. Chang and J. Edmonds)

7.14 The Payment Scheduling Problem Let G = $(\mathcal{N}, \mathcal{A})$ be the arrow diagram for a project with nodes 1, n as the start and finish nodes, and $(t_{ij} : (i, j) \in \mathcal{A})$ as the vector of job durations. Define t_1 = 0, t_i for $i = 2$ to n to be the time when the event corresponding to node i is realized. In this problem, the variables are t_2 to t_n, the vector $t = (t_2, \ldots, t_n)$ is called the *schedule*. The realization of the event corresponding to any node i is usually the occasion for a transaction (either the contractor doing the project is given a stage payment for reaching this milestone in this project, or he may have to pay a subcontractor whom he hired to do part of the work), let c_i (> 0 for receipts, < 0 for payments) denote the cash reward to the contractor at this event. The payment scheduling problem is concerned with finding a feasible schedule that maximizes the present value (at the commencement of the project) of all the cash transactions. Assuming that the discount rate for money per unit per unit time is β, this is the problem of finding a schedule t that maximizes $P(t) = \sum(c_i \exp(-\beta t_i)$: over $i = 2$ to $n)$, subject to $t_j - t_i \geqq t_{ij}$ for each $(i, j) \in \mathcal{A}$ and $t_n - t_1 \leqq \lambda$ $=$ upper bound on project completion time. Show that this problem can be transformed into an LP by transforming the variables using $t_i = -(1/\beta) \log(y_i), i = 2$ to n.

Show that every extreme point of this LP corresponds to a spanning tree in $\mathrm{G}' = (\mathcal{N}, \mathcal{A}')$ where $\mathcal{A}' = \mathcal{A} \cup \{(n, 1)\}$, and vice versa. Hence the search for optimal schedules can be restricted to feasible trees in G'. Using standard complementary slackness results for checking the optimality of feasible trees, develop a simplex -like procedure for this problem that moves from one feasible tree to an adjacent one obtained by changing the tree by one arc, improving the objective function value in each move, until an optimum schedule is obtained. (Grinold [1972], Russel [1970]. See Elmaghraby and Herroelen [1990] for a critique of this model.)

Comment 7.1 The first paper to discuss the problem of computing the cost curves for a project composed of many individual jobs is that by Kelly and Walker [1959]. In this chapter we discussed methods for computing the project cost curve by network flow methods due to Fulkerson [1961] and Kelly [1961]. Since the appearance of these papers, the network-based CPM has become a part of the language of project management, and has been used extensively in 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, 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 OR software. In the chapter's exercises, we have included some modeling problems taken from simplified real world applications.

The literature on network techniques for project management is enormous. Battersby [1967], Burman [1972], Elmaghraby [1977], Weist and Levy [1977] are some of the books devoted exclusively to this area.

Krishnamoorthy and Deo [1979] are the first to show that the problem of generating an arrow diagram for a project using the smallest number of dummy arcs is NP-hard. The papers of Syslo [1981, 1984] explore some other complexity issues associated with arrow diagrams. Dimsdale [1963], Fisher, Liebman, and Nemhauser [1968], discuss practical techniques for generating arrow diagrams.

The papers of Russel [1970], and Grinold [1972] discuss the payment scheduling problem. Elmaghraby and Herroelen [1990] give a critique of this model.

7.6 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. G. CORNEIL, C. C. GOTLIEB and Y. M. LEE, 1973, "Minimal Event-Node Network of Project Precedence relations," *CACM*, 16(296-298).

D. DIMSDALE, March 1963, "Computer Construction of Minimal Project Network," *IBM Systems Journal*, 2(24-36).

S. E. ELMAGHRABY, 1977, *Activity Networks*, Wiley, NY.

S. E. ELMAGHRABY and W. S. HERROELEN, 1990, "The Scheduling of Activities to Maximize the Net Present Value of Projects," *EJOR*, 49(35-49).

A. C. FISHER, D. S. LIEBMAN, and G. L. NEMHAUSER, July 1968, "Computer Construction of Project Networks," *CACM*, 11(493-497).

D. R. FULKERSON, Jan. 1961, "A Network Flow Computation for Project Cost Curve," *MS*, 7, no. 2 (167-178).

R. C. GRINOLD, 1972, "The Payment Scheduling Problem," *NRLQ*, 19, no. 1(123-136).

A. KANDA and V. R. K. RAO, May 1984, "A Network Flow Procedure for Project

Crashing with Penalty Nodes," *EJOR*, 16, n0. 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," *OR*, 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.

M. S. KRISHNAMOORTHY and N. DEO, 1979, "Complexity of the Minimum Dummy Activities Problem in a PERT Network," *Networks*, 9(189-194).

A. H. RUSSEL, Jan. 1970, "Cash Flows in Networks," *MS*, 16, no. 5(357-373).

M. M. SYSLO, 1981, "Optimal Constructions of Event-Node Networks," *RAIRO Recherche Operationelle*, 15(241-260).

M. M. SYSLO, 1984, "On the Computational Complexity of the Minimum-Dummy-Activities Problem in a PERT Network," *Networks*, 14(37-45).

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," *ORQ*, 27, no. 2, i(341-349).

Index

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

AOA diagram 547
AON diagram 547
Activities 544
Ancestor 545
Arrow diagram 547

Backward label 555
Backward pass 555

Crashing 559
Critical 554
  Activities 554
  Arcs 554
  Jobs 554
  Path 557

Descendent 545
Dummy 548
  Arcs 548
  Jobs 548

Early 554
  Finish 554
  Start 554
Events 547

Finite breakthrough 565
Forward 554, 555
  Label 555
  Pass 554
Free 557
  Float 557
  Slack 557

∞-breakthrough 565

Job 544
  Shortening 559

Late 554
  Finish 554
  Start 554

Min. project duration 553

PERT 573
Partial ordering 544
Predecessor 545
  Immediate 545
Project network 544
Project scheduling 553
  Resource constrained 572

Redundancy 546

Safety float 556
Successor 545
    Immediate 545

Total float 556
Total slack 556
Transitivity 545