Fundamental Scheduling Procedures

10.1 Relevance of Construction Schedules

In addition to assigning dates to project activities, project scheduling is intended to match the resources of equipment, materials and labor with project work tasks over time. Good scheduling can eliminate problems due to production bottlenecks, facilitate the timely procurement of necessary materials, and otherwise insure the completion of a project as soon as possible. In contrast, poor scheduling can result in considerable waste as laborers and equipment wait for the availability of needed resources or the completion of preceding tasks. Delays in the completion of an entire project due to poor scheduling can also create havoc for owners who are eager to start using the constructed facilities.

Attitudes toward the formal scheduling of projects are often extreme. Many owners require detailed construction schedules to be submitted by contractors as a means of monitoring the work progress. The actual work performed is commonly compared to the schedule to determine if construction is proceeding satisfactorily. After the completion of construction, similar comparisons between the planned schedule and the actual accomplishments may be performed to allocate the liability for project delays due to changes requested by the owner, worker strikes or other unforeseen circumstances.

In contrast to these instances of reliance upon formal schedules, many field supervisors disdain and dislike formal scheduling procedures. In particular, the *critical path method* of scheduling is commonly required by owners and has been taught in universities for over two decades, but is often regarded in the field as irrelevant to actual operations and a time consuming distraction. The result is "seat-of-the-pants" scheduling that can be good or that can result in grossly inefficient schedules and poor productivity. Progressive construction firms use formal scheduling procedures whenever the complexity of work tasks is high and the coordination of different workers is required.

Formal scheduling procedures have become much more common with the advent of personal computers on construction sites and easy-to-use software programs. Sharing schedule information via the Internet has also provided a greater incentive to use formal scheduling methods. Savvy construction supervisors often carry schedule and budget information around with wearable or handheld computers. As a result, the continued development of easy to use computer programs and improved methods of presenting schedules hav overcome the practical problems associated with formal scheduling mechanisms. But problems with the use of scheduling techniques will continue until managers understand their proper use and limitations.

A basic distinction exists between *resource oriented* and *time oriented* scheduling techniques. For resource oriented scheduling, the focus is on using and scheduling particular resources in an effective fashion. For example, the project manager's main concern on a high-rise building site might be to insure that cranes are used effectively for moving materials; without effective scheduling in this case, delivery trucks might queue on the ground and workers wait for deliveries on upper floors. For time oriented scheduling, the emphasis is on determining the completion time of the project given the necessary precedence relationships among activities. Hybrid techniques for resource leveling or resource constrained scheduling in the presence of precedence relationships also exist. Most scheduling software is time-oriented, although virtually all of the programs have the capability to introduce resource constaints.

This chapter will introduce the fundamentals of scheduling methods. Our discussion will generally assume that computer based scheduling programs will be applied. Consequently, the wide variety of manual or mechanical scheduling techniques will not be discussed in any detail. These manual methods are not as capable or as convenient as computer based scheduling. With the availability of these computer based scheduling programs, it is important for managers to understand the basic operations performed by scheduling programs. Moreover, even if formal methods are not applied in particular cases, the conceptual framework of formal scheduling methods provides a valuable reference for a manager. Accordingly, examples involving hand calculations will be provided throughout the chapter to facilitate understanding.

Back to top

10.2 The Critical Path Method

The most widely used scheduling technique is the critical path method (CPM) for scheduling, often referred to as *critical path scheduling*. This method calculates the minimum completion time for a project along with the possible start and finish times for the project activities. Indeed, many texts and managers regard critical path scheduling as the only usable and practical scheduling procedure. Computer programs and algorithms for critical path scheduling are widely available and can efficiently handle projects with thousands of activities.

The *critical path* itself represents the set or sequence of predecessor/successor activities which will take the longest time to complete. The duration of the critical path is the sum of the activities' durations along the path. Thus, the critical path can be defined as the longest possible path through the "network" of project activities, as described in Chapter 9. The duration of the critical path represents the minimum time required to complete a project. Any delays along the critical path would imply that additional time would be required to complete the project.

There may be more than one critical path among all the project activities, so completion of the entire project could be delayed by delaying activities along any one of the critical paths. For example, a project consisting of two activities performed in parallel that each require three days would have each activity critical for a completion in three days.

Formally, critical path scheduling assumes that a project has been divided into activities of fixed duration and well defined predecessor relationships. A predecessor relationship implies that one activity must come before another in the schedule. No resource constraints other than those implied by precedence relationships are recognized in the simplest form of critical path scheduling.

To use critical path scheduling in practice, construction planners often represent a *resource constraint* by a precedence relation. A *constraint* is simply a restriction on the options available to a manager, and a *resource constraint* is a constraint deriving from the limited availability of some resource of equipment, material, space or labor. For example, one of two activities requiring the same piece of equipment might be arbitrarily assumed to precede the other activity. This artificial precedence constraint insures that the two activities requiring the same resource will not be scheduled at the same time. Also, most critical path scheduling algorithms impose restrictions on the generality of the activity relationships or network geometries which are used. In essence, these restrictions imply that the construction plan can be represented by a network plan in which activities appear as nodes in a network, as in Figure 9-6. Nodes are numbered, and no two nodes can have the same number or designation. Two nodes are introduced to represent the start and completion of the project itself.

The actual computer representation of the project schedule generally consists of a list of activities along with their associated durations, required resources and predecessor activities. Graphical network representations rather than a list are helpful for visualization of the plan and to insure that mathematical requirements are met. The actual input of the data to a computer program may be accomplished by filling in blanks on a screen menu, reading an existing datafile, or typing data directly to the program with identifiers for the type of information being provided.

With an activity-on-branch network, dummy activities may be introduced for the purposes of providing unique activity designations and maintaining the correct sequence of activities. A *dummy activity* is assumed to have no time duration and can be graphically represented by a dashed line in a network. Several cases in which dummy activities are useful are illustrated in Fig. 10-1. In Fig. 10-1(a), the elimination of activity C would mean that both activities B and D would be identified as being between nodes 1 and 3. However, if a dummy activity X is introduced, as shown in part (b) of the figure, the unique designations for activity B (node 1 to 2) and D (node 1 to 3) will be preserved. Furthermore, if the problem in part (a) is changed so that activity E cannot start until both C and D are completed but that F can start after D alone is completed, the order in the new sequence can be indicated by the addition of a dummy activity Y, as shown in part (c). In general, dummy activities may be necessary to meet the requirements of specific computer scheduling algorithms, but it is important to limit the number of such dummy link insertions to the extent possible.

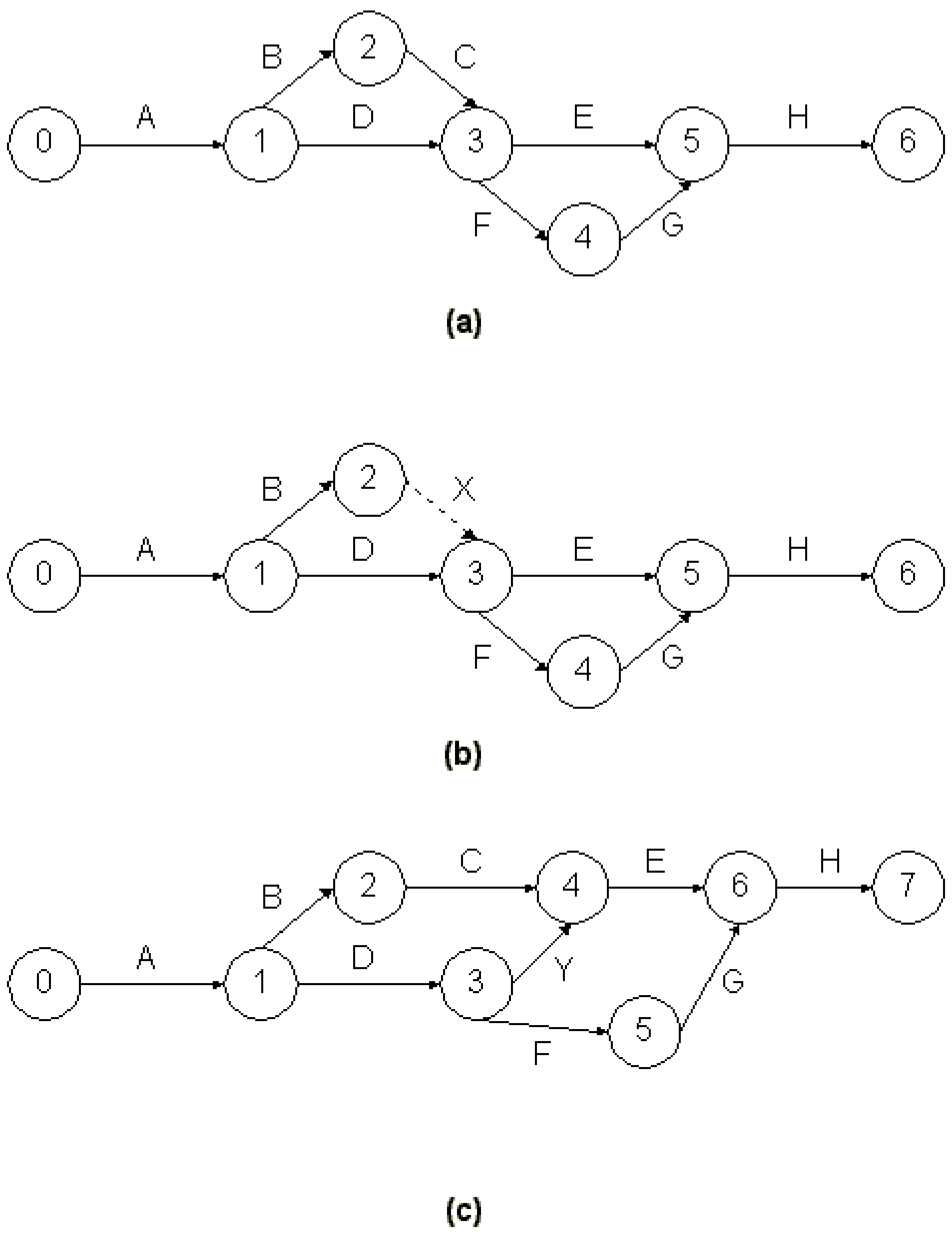


Figure 10-1 Dummy Activities in a Project Network

Many computer scheduling systems support only one network representation, either activity-on-branch or acitivity-on-node. A good project manager is familiar with either representation.

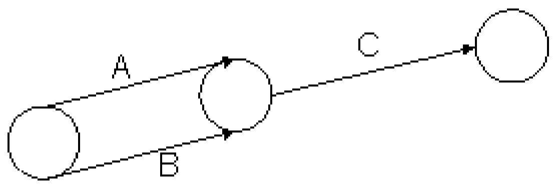
Example 10-1: Formulating a network diagram

Suppose that we wish to form an activity network for a seven-activity network with the following precedences:

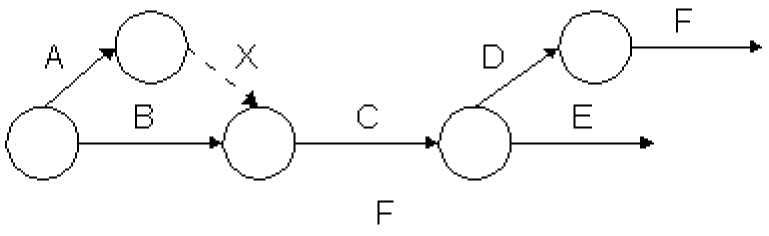
Activity	**Predecessors**

A	---
B	---
C	A,B
D	C
E	C
F	D
G	D,E

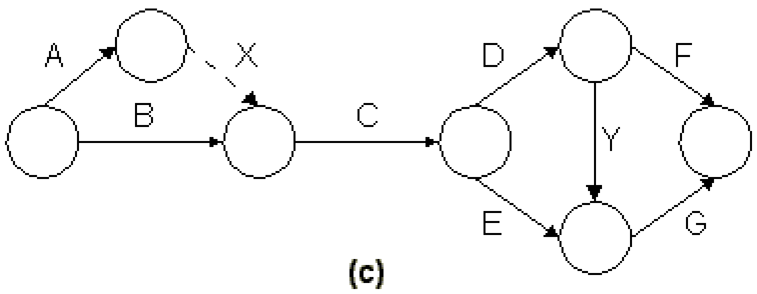
Forming an activity-on-branch network for this set of activities might begin be drawing activities A, B and C as shown in Figure 10-2(a). At this point, we note that two activities (A and B) lie between the same two event nodes; for clarity, we insert a dummy activity X and continue to place other activities as in Figure 10-2(b). Placing activity G in the figure presents a problem, however, since we wish both activity D and activity E to be predecessors. Inserting an additional dummy activity Y along with activity G completes the activity network, as shown in Figure 10-2(c). A comparable activity-on-node representation is shown in Figure 10-3, including project start and finish nodes. Note that dummy activities are not required for expressing precedence relationships in activity-on-node networks.



(a)



(b)



(c)

Figure 10-2 An Activity-on-Branch Network for Critical Path Scheduling

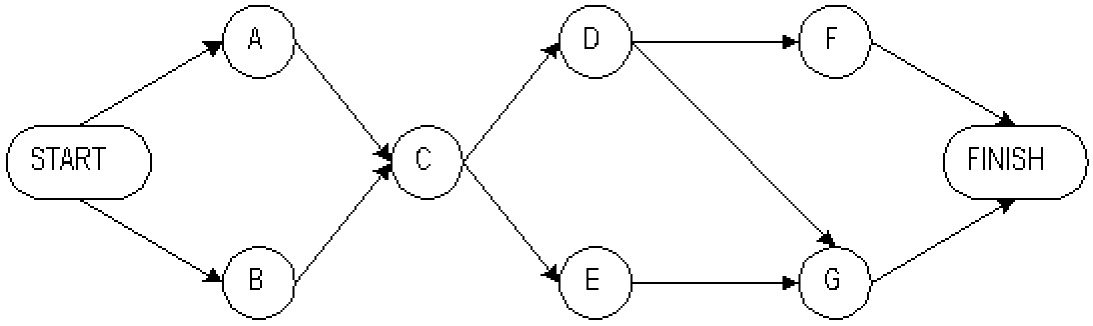


Figure 10-3 An Activity-on-Node Network for Critical Path Scheduling

Back to top

10.3 Calculations for Critical Path Scheduling

With the background provided by the previous sections, we can formulate the critical path scheduling mathematically. We shall present an algorithm or set of instructions for critical path scheduling assuming an activity-on-branch project network. We also assume that all precedences are of a finish-to-start nature, so that a succeeding activity cannot start until the completion of a preceding activity. In a later section, we present a comparable algorithm for activity-on-node representations with multiple precedence types.

Suppose that our project network has n+1 nodes, the initial event being 0 and the last event being n. Let the time at which node events occur be x_1, x_2,...., x_n, respectively. The start of the project at x_0 will be defined as time 0. Nodal event times must be consistent with activity durations, so that an activity's successor node event time must be larger than an activity's predecessor node event time plus its duration. For an activity defined as starting from event i and ending at event j, this relationship can be expressed as the inequality constraint, $x_j \geq x_i + D_{ij}$ where D_{ij} is the duration of activity (i,j). This same expression can be written for every activity and must hold true in any feasible schedule. Mathematically, then, the critical path scheduling problem is to minimize the time of project completion (x_n) subject to the constraints that each node completion event cannot occur until each of the predecessor activities have been completed:

Minimize
(10.1)

$$z = x_n$$

subject to

$$x_0 = 0$$

$$x_j - x_i - D_{ij} \geq 0 \quad for\ each\ activity\ (i,j)$$

This is a linear programming problem since the objective value to be minimized and each of the constraints is a linear equation. [1]

Rather than solving the critical path scheduling problem with a linear programming algorithm (such as the Simplex method), more efficient techniques are available that take advantage of the network structure of the problem. These solution methods are very efficient with respect to the required computations, so that very large networks can be treated even with personal computers. These methods also give some very useful information about possible activity schedules. The programs can compute the earliest and latest possible starting times for each activity which are consistent with completing the project in the shortest possible time. This calculation is of particular interest for activities which are not on the critical path (or paths), since these activities might be slightly delayed or re-scheduled over time as a manager desires without delaying the entire project.

An efficient solution process for critical path scheduling based upon node labeling is shown in Table 10-1. Three algorithms appear in the table. The *event numbering algorithm* numbers the nodes (or events) of the project such that the beginning event has a lower number than the ending event for each activity. Technically, this algorithm accomplishes a "topological sort" of the activities. The project start node is given number 0. As long as the project activities fulfill the conditions for an activity-on-branch network, this type of numbering system is always possible. Some software packages for critical path scheduling do not have this numbering algorithm programmed, so that the construction project planners must insure that appropriate numbering is done.

TABLE 10-1 Critical Path Scheduling Algorithms (Activity-on-Branch Representation)

Event Numbering Algorithm

Step 1: Give the starting event number 0.
Step 2: Give the next number to any unnumbered event whose predecessor events are each already numbered.
Repeat Step 2 until all events are numbered.

Earliest Event Time Algorithm

Step 1: Let E(0) = 0.
Step 2: For j = 1,2,3,...,n (where n is the last event), let
E(j) = maximum {E(i) + D_{ij}}
where the maximum is computed over all activities (i,j) that have j as the ending event.

Latest Event Time Algorithm

Step 1: Let L(n) equal the required completion time of the project.
Note: L(n) must equal or exceed E(n).
Step 2: For i = n-1, n-2, ..., 0, let
L(i) = minimum {L(j) - D_{ij}}
where the minimum is computed over all activities (i,j) that have i as the starting event.

The *earliest event time algorithm* computes the earliest possible time, E(i), at which each event, i, in the network can occur. Earliest event times are computed as the maximum of the earliest start times plus activity durations for each of the activities immediately preceding an event. The earliest start time for each activity (i,j) is equal to the earliest possible time for the preceding event E(i):

(10.2)
$$ES(i,j) = E(i)$$

The earliest finish time of each activity (i,j) can be calculated by:

(10.3)
$$EF(i,j) = E(i) + D_{ij}$$

Activities are identified in this algorithm by the predecessor node (or event) i and the successor node j. The algorithm simply requires that each event in the network should be examined in turn beginning with the project start (node 0).

The *latest event time algorithm* computes the latest possible time, L(j), at which each event j in the network can occur, given the desired completion time of the project, L(n) for the last event n. Usually, the desired completion time will be equal to the earliest possible completion time, so that E(n) = L(n) for the final node n. The procedure for finding the latest event time is analogous to that for the earliest event time except that the procedure begins with the final event and works backwards through the project activities. Thus, the earliest event time algorithm is often called a *forward pass* through the network, whereas the latest event time algorithm is the the *backward pass* through the network. The latest finish time consistent with completion of the project in the desired time frame of L(n) for each activity (i,j) is equal to the latest possible time L(j) for the succeeding event:

(10.4) $$LF(i,j) = L(j)$$

The latest start time of each activity (i,j) can be calculated by:

(10.5) $$LS(i,j) = L(j) - D_{ij}$$

The earliest start and latest finish times for each event are useful pieces of information in developing a project schedule. Events which have equal earliest and latest times, E(i) = L(i), lie on the critical path or paths. An activity (i,j) is a critical activity if it satisfies all of the following conditions:

(10.6) $$E(i) = L(i)$$

(10.7) $$E(j) = L(j)$$

(10.8) $$E(i) + D_{ij} = L(j)$$

Hence, activities between critical events are also on a critical path as long as the activity's earliest start time equals its latest start time, ES(i,j) = LS(i,j). To avoid delaying the project, all the activities on a critical path should begin as soon as possible, so each critical activity (i,j) must be scheduled to begin at the earliest possible start time, E(i).

Example 10-2: Critical path scheduling calculations

Consider the network shown in Figure 10-4 in which the project start is given number 0. Then, the only event that has each predecessor numbered is the successor to activity A, so it receives number 1. After this, the only event that has each predecessor numbered is the successor to the two activities B and C, so it receives number 2. The other event numbers resulting from the algorithm are also shown in the figure. For this simple project network, each stage in the numbering process found only one possible event to number at any time. With more than one feasible event to number, the choice of which to number next is arbitrary. For example, if activity C did not exist in the project for Figure 10-4, the successor event for activity A or for activity B could have been numbered 1.

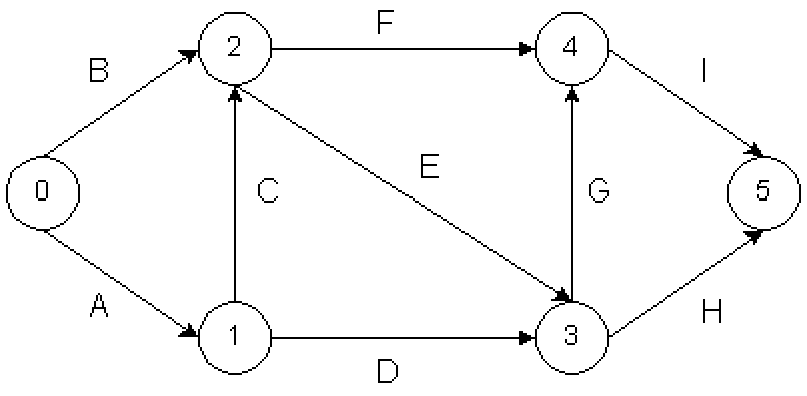


Figure 10-4 A Nine-Activity Project Network

Once the node numbers are established, a good aid for manual scheduling is to draw a small rectangle near each node with two possible entries. The left hand side would contain the earliest time the event could occur, whereas the right hand side would contain the latest time the event could occur without delaying the entire project. Figure 10-5 illustrates a typical box.

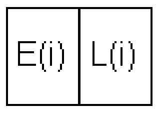


Figure 10-5 E(i) and L(i) Display for Hand Calculation of Critical Path for Activity-on-Branch Representation

TABLE 10-2 Precedence Relations and Durations for a Nine Activity Project Example

Activity	Description	Predecessors	Duration
A	Site clearing	---	4
B	Removal of trees	---	3
C	General excavation	A	8
D	Grading general area	A	7
E	Excavation for trenches	B, C	9

F	Placing formwork and reinforcement for concrete	B, C	12
G	Installing sewer lines	D, E	2
H	Installing other utilities	D, E	5
I	Pouring concrete	F, G	6

For the network in Figure 10-4 with activity durations in Table 10-2, the earliest event time calculations proceed as follows:

Step 1 $\rightarrow$ E(0) = 0

Step 2

j = 1 $\rightarrow$ $E(1) = Max\{E(0) + D_{01}\} = Max\{ 0 + 4 \} = 4$

j = 2 $\rightarrow$ $E(2) = Max\{E(0) + D_{02}; E(1) + D_{12}\} = Max\{0 + 3; 4 + 8\} = 12$

j = 3 $\rightarrow$ $E(3) = Max\{E(1) + D_{13}; E(2) + D_{23}\} = Max\{4 + 7; 12 + 9\} = 21$

j = 4 $\rightarrow$ $E(4) = Max\{E(2) + D_{24}; E(3) + D_{34}\} = Max\{12 + 12; 21 + 2\} = 24$

j = 5 $\rightarrow$ $E(5) = Max\{E(3) + D_{35}; E(4) + D_{45}\} = Max\{21 + 5; 24 + 6\} = 30$

Thus, the minimum time required to complete the project is 30 since E(5) = 30. In this case, each event had at most two predecessors.

For the "backward pass," the latest event time calculations are:

Step 1 $\rightarrow$ L(5) = E(5) = 30

Step 2

j = 4 $\rightarrow$ $L(4) = Min \{L(5) - D_{45}\} = Min \{30 - 6\} = 24$

j = 3 $\rightarrow$ $L(3) = Min \{L(5) - D_{35}; L(4) - D_{34}\} = Min \{30 -5; 24 - 2\} = 22$

j = 2 $\rightarrow$ $L(2) = Min \{L(4) - D_{24}; L(3) - D_{23}\} = Min \{24 - 12; 22 - 9\} = 12$

j = 1 $\rightarrow$ $L(1) = Min \{L(3) - D_{13}; L(2) - D_{12}\} = Min \{22 - 7; 12 - 8\} = 4$

j = 0 $\rightarrow$ $L(0) = Min \{L(2) - D_{02}; L(1) - D_{01}\} = Min \{12 - 3; 4 - 4\} = 0$

In this example, E(0) = L(0), E(1) = L(1), E(2) = L(2), E(4) = L(4),and E(5) = L(5). As a result, all nodes but node 3 are in the critical path. Activities on the critical path include A (0,1), C (1,2), F (2,4) and I (4,5) as shown in Table 10-3.

TABLE 10-3 Identification of Activities on the Critical Path for a Nine-Activity Project

Activity	Duration D_{ij}	Earliest start time E(i)=ES(i,j)	Latest finish time L(j)=LF(i,j)	Latest start time LS(i,j)
A (0,1)	4	0*	4*	0
B (0,2)	3	0	12	9
C (1,2)	8	4*	12*	4
D (1,3)	7	4	22	15
E (2,3)	9	12	22	13
F (2,4)	12	12*	24*	12

G (3,4) H (3,5) I (4,5)	2 5 6	21 21 24	24 30 30*	22 25 24

*Activity on a critical path since E(i) + D_{iJ} = L(j).

Back to top

10.4 Activity Float and Schedules

A number of different activity schedules can be developed from the critical path scheduling procedure described in the previous section. An *earliest time* schedule would be developed by starting each activity as soon as possible, at ES(i,j). Similarly, a *latest time* schedule would delay the start of each activity as long as possible but still finish the project in the minimum possible time. This late schedule can be developed by setting each activity's start time to LS(i,j).

Activities that have different early and late start times (i.e., ES(i,j) < LS(i,j)) can be scheduled to start anytime between ES(i,j) and LS(i,j) as shown in Figure 10-6. The concept of *float* is to use part or all of this allowable range to schedule an activity without delaying the completion of the project. An activity that has the earliest time for its predecessor and successor nodes differing by more than its duration possesses a window in which it can be scheduled. That is, if E(i) + D_{ij} < L(j), then some float is available in which to schedule this activity.

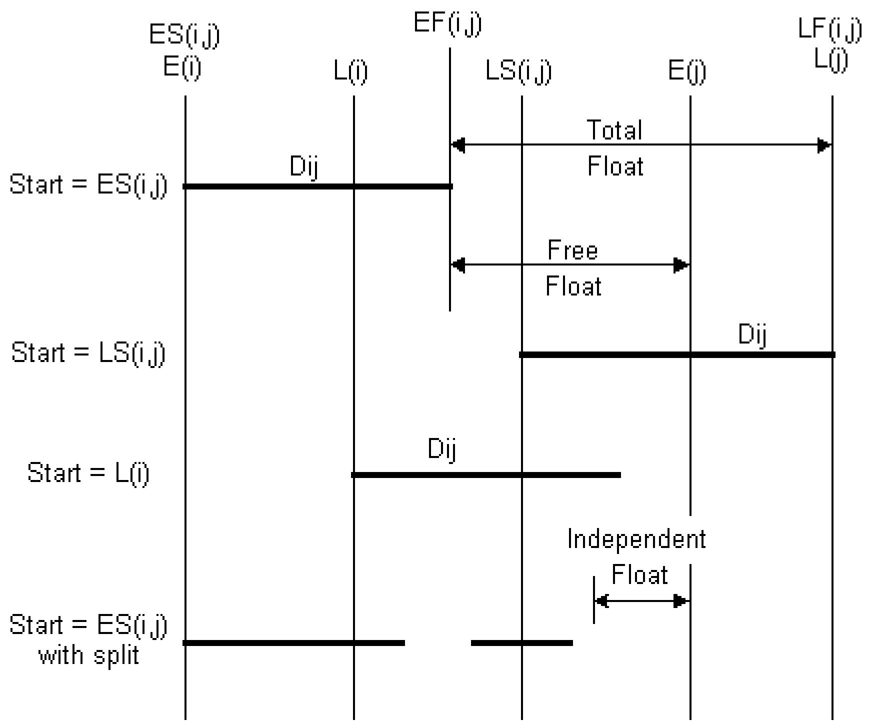


Figure 10-6 Illustration of Activity Float

Float is a very valuable concept since it represents the scheduling flexibility or "maneuvering room" available to complete particular tasks. Activities on the critical path do not provide any flexibility for scheduling nor leeway in case of problems. For activities with some float, the actual starting time might be chosen to balance work loads over time, to correspond with material deliveries, or to improve the project's cash flow.

Of course, if one activity is allowed to float or change in the schedule, then the amount of float available for other activities may decrease. Three separate categories of float are defined in critical path scheduling:

1. *Free float* is the amount of delay which can be assigned to any one activity without delaying subsequent activities. The free float, FF(i,j), associated with activity (i,j) is:

   (10.9) $$FF(i,j) = E(j) - E(i) - D_{ij}$$

2. *Independent float* is the amount of delay which can be assigned to any one activity without delaying subsequent activities or restricting the scheduling of preceding activities. Independent float, IF(i,j), for activity (i,j) is calculated as:

   (10.10) $$IF(i,j) = \begin{cases} 0 \\ E(j) - L(i) - D_{ij} \end{cases}$$

3. *Total float* is the maximum amount of delay which can be assigned to any activity without delaying the entire project. The total float, TF(i,j), for any activity (i,j) is calculated as:

   (10.11) $$TF(i,j) = L(j) - E(i) - D_{ij}$$

Each of these "floats" indicates an amount of flexibility associated with an activity. In all cases, total float equals or exceeds free float, while independent float is always less than or equal to free float. Also, any activity on a critical path has all three values of float equal to zero. The converse of this statement is also true, so any activity which has zero total float can be recognized as being on a critical path.

The various categories of activity float are illustrated in Figure 10-6 in which the activity is represented by a bar which can move back and forth in time depending upon its scheduling start. Three possible scheduled starts are shown, corresponding to the cases of starting each activity at the earliest event time, E(i), the latest activity start time LS(i,j), and at the latest event time L(i). The three categories of float can be found directly from this figure. Finally, a fourth bar is included in the figure to illustrate the possibility that an activity might start, be temporarily halted, and then re-start. In this case, the temporary halt was sufficiently short that it was less than the independent float time and thus would not interfere with other activities. Whether or not such work splitting is possible or economical depends upon the nature of the activity.

As shown in Table 10-3, activity D(1,3) has free and independent floats of 10 for the project shown in Figure 10-4. Thus, the start of this activity could be scheduled anytime between time 4 and 14 after the project began without interfering with the schedule of other activities or with the earliest completion time of the project. As the total float of 11 units indicates, the start of activity D could also be delayed until time 15, but this would require that the schedule of other activities be restricted. For example, starting activity D at time 15 would require that activity G would begin as soon as activity D was completed. However, if this schedule was maintained, the overall completion date of the project would not be changed.

Example 10-3: Critical path for a fabrication project

As another example of critical path scheduling, consider the seven activities associated with the fabrication of a steel component shown in Table 10-4. Figure 10-7 shows the network diagram associated with these seven activities. Note that an additional dummy activity X has been added to insure that the correct precedence relationships are maintained for activity E. A simple rule to observe is that if an activity has more than one immediate predecessor and another activity has at least one but not all of these predecessor activity as a predecessor, a dummy activity will be required to maintain precedence relationships. Thus, in the figure, activity E has activities B and C as predecessors, while activity D has only activity C as a predecessor. Hence, a dummy activity is required. Node numbers have also been added to this figure using the procedure outlined in Table 10-1. Note that the node numbers on nodes 1 and 2 could have been exchanged in this numbering process since after numbering node 0, either node 1 or node 2 could be numbered next.

TABLE 10-4 Precedences and Durations for a Seven Activity Project

Activity	Description	Predecessors	Duration
A	Preliminary design	---	6
B	Evaluation of design	A	1
C	Contract negotiation	---	8
D	Preparation of fabrication plant	C	5
E	Final design	B, C	9
F	Fabrication of Product	D, E	12
G	Shipment of Product to owner	F	3

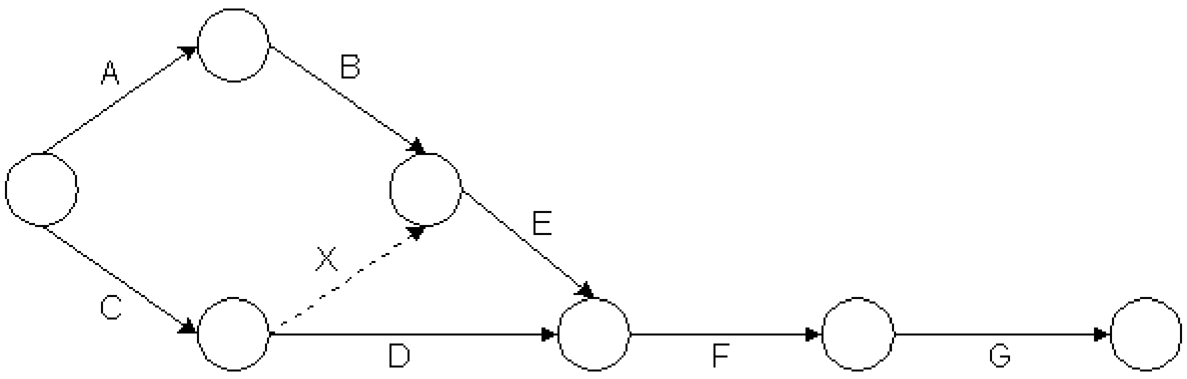


Figure 10-7 Illustration of a Seven Activity Project Network

The results of the earliest and latest event time algorithms (appearing in Table 10-1) are shown in Table 10-5. The minimum completion time for the project is 32 days. In this small project, all of the event nodes except node 1 are on the critical path. Table 10-6 shows the earliest and latest start times for the various activities including the different categories of float. Activities C,E,F,G and the dummy activity X are seen to lie on the critical path.

TABLE 10-5 Event Times for a Seven Activity Project

Node	Earliest Time E(i)	Latest Time L(j)
0	0	0
1	6	7
2	8	8
3	8	8
4	17	17
5	29	29
6	32	32

TABLE 10-6 Earliest Start, Latest Start and Activity Floats for a Seven Activity Project

Activity	Earliest start time	Latest start time ES(i,j)	Free float LS(i,j)	Independent float	Total float
A (0,1)	0	1	0	0	1
B (1,3)	6	7	1	0	1
C (0,2)	0	0	0	0	0
D (2,4)	8	12	4	4	4
E (3,4)	8	8	0	0	0
F (4,5)	17	17	0	0	0
G (5,6)	29	29	0	0	0
X (2,3)	8	8	0	0	0

Back to top

10.5 Presenting Project Schedules

Communicating the project schedule is a vital ingredient in successful project management. A good presentation will greatly ease the manager's problem of understanding the multitude of activities and their inter-relationships. Moreover, numerous individuals and parties are involved in any project, and they have to understand their assignments. *Graphical* presentations of project schedules are particularly useful since it is much easier to comprehend a graphical display of numerous pieces of information than to sift through a large table of numbers. Early computer scheduling systems were particularly poor in this regard since they produced pages and pages of numbers without aids to the manager for understanding them. A short example appears in Tables 10-5 and 10-6; in practice, a project summary table would be much longer. It is extremely tedious to read a table of activity numbers, durations, schedule times, and floats and thereby gain an understanding and appreciation of a project schedule. In practice, producing diagrams manually has been a common prescription to the lack of automated drafting facilities. Indeed, it has been common to use computer programs to perform critical path scheduling and then to produce bar charts of detailed activity schedules and resource assignments manually. With the availability of computer graphics, the cost and effort of producing graphical presentations has been significantly reduced and the production of presentation aids can be automated.

Network diagrams for projects have already been introduced. These diagrams provide a powerful visualization of the precedences and relationships among the various project activities. They are a basic means of communicating a project plan among the participating planners and project monitors. Project planning is often conducted by producing network representations of greater and greater refinement until the plan is satisfactory.

A useful variation on project network diagrams is to draw a *time-scaled* network. The activity diagrams shown in the previous section were topological networks in that only the relationship between nodes and branches were of interest. The actual diagram could be distorted in any way desired as long as the connections between nodes were not changed. In time-scaled network diagrams, activities on the network are plotted on a horizontal axis measuring the time since project commencement. Figure 10-8 gives an example of a time-scaled activity-on-branch diagram for the nine activity project in Figure 10-4. In this time-scaled diagram, each node is shown at its earliest possible time. By looking over the horizontal axis, the time at which activity can begin can be observed. Obviously, this time scaled diagram is produced as a display after activities are initially scheduled by the critical path method.

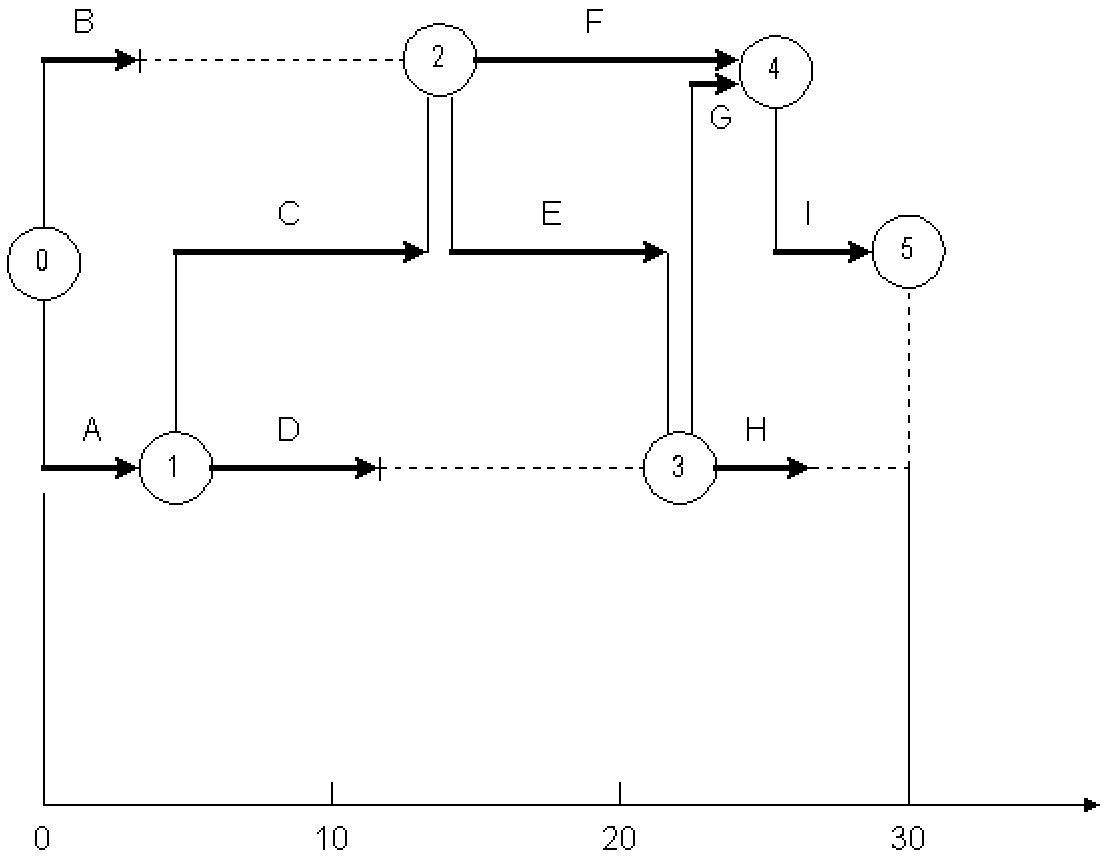


Figure 10-8 Illustration of a Time Scaled Network Diagram with Nine Activities

Another useful graphical representation tool is a bar or Gantt chart illustrating the scheduled time for each activity. The bar chart lists activities and shows their scheduled start, finish and duration. An illustrative bar chart for the nine activity project appearing in Figure 10-4 is shown in Figure 10-9. Activities are listed in the vertical axis of this figure, while time since project commencement is shown along the horizontal axis. During the course of *monitoring* a project, useful additions to the basic bar chart include a vertical line to indicate the current time plus small marks to indicate the current state of work on each activity. In Figure 10-9, a hypothetical project state after 4 periods is shown. The small "v" marks on each activity represent the current state of each activity.

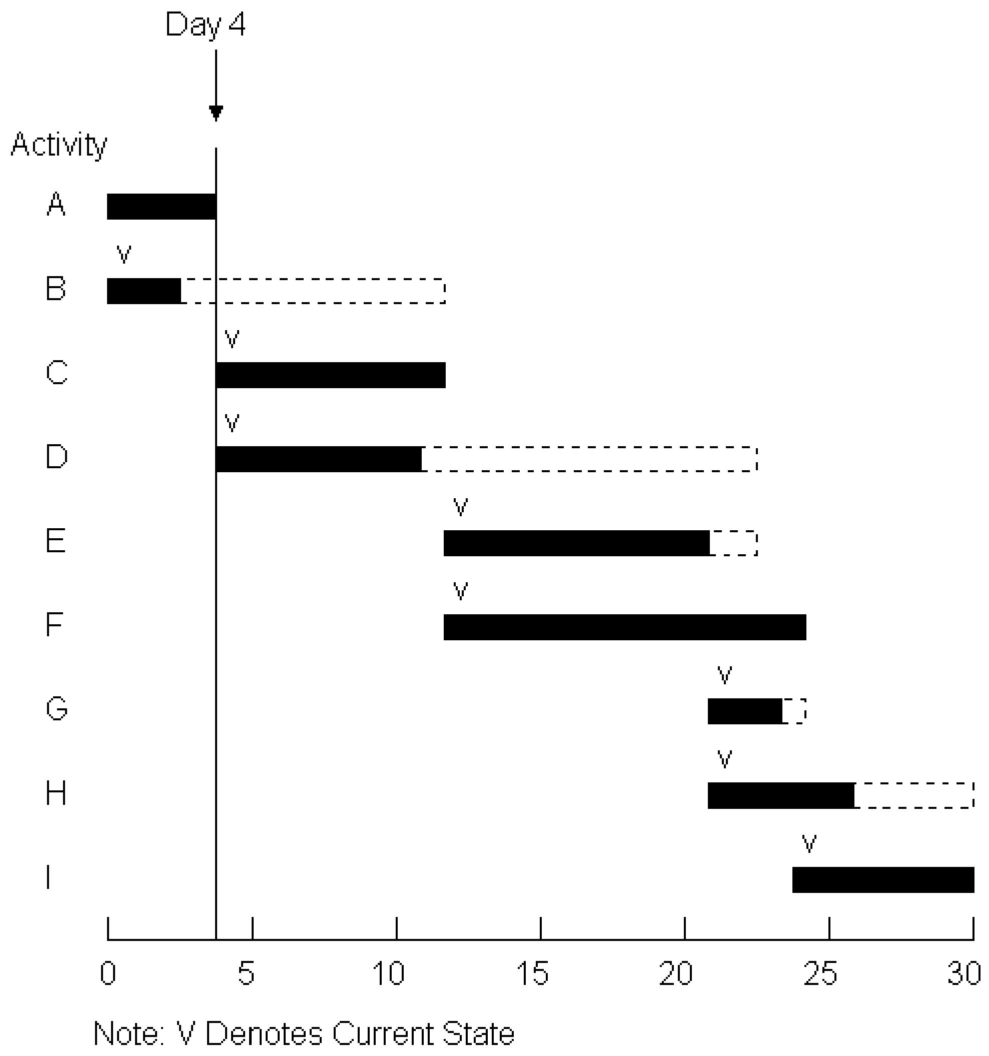


Figure 10-9 An Example Bar Chart for a Nine Activity Project

Bar charts are particularly helpful for communicating the current state and schedule of activities on a project. As such, they have found wide acceptance as a project representation tool in the field. For planning purposes, bar charts are not as useful since they do not indicate the precedence relationships among activities. Thus, a planner must remember or record separately that a change in one activity's schedule may require changes to successor activities. There have been various schemes for mechanically linking activity bars to represent precedences, but it is now easier to use computer based tools to represent such relationships.

Other graphical representations are also useful in project monitoring. Time and activity graphs are extremely useful in portraying the current status of a project as well as the existence of activity float. For example, Figure 10-10 shows two possible schedules for the nine activity project described in Table 9-1 and shown in the previous figures. The first schedule would occur if each activity was scheduled at its earliest start time, ES(i,j) consistent with completion of the project in the minimum possible time. With this schedule, Figure 10-10 shows the percent of project activity completed versus time. The second schedule in Figure 10-10 is based on latest possible start times for each activity, LS(i,j). The horizontal time difference between the two feasible schedules gives an indication of the extent of possible float. If the project goes according to plan, the actual percentage completion at different times should fall between these curves. In practice, a vertical axis representing cash expenditures rather than percent completed is often used in developing a project representation of this type. For this purpose, activity cost estimates are used in preparing a time versus completion graph. Separate "S-curves" may also be prepared for groups of activities on the same graph, such as separate curves for the design, procurement, foundation or particular sub-contractor activities.

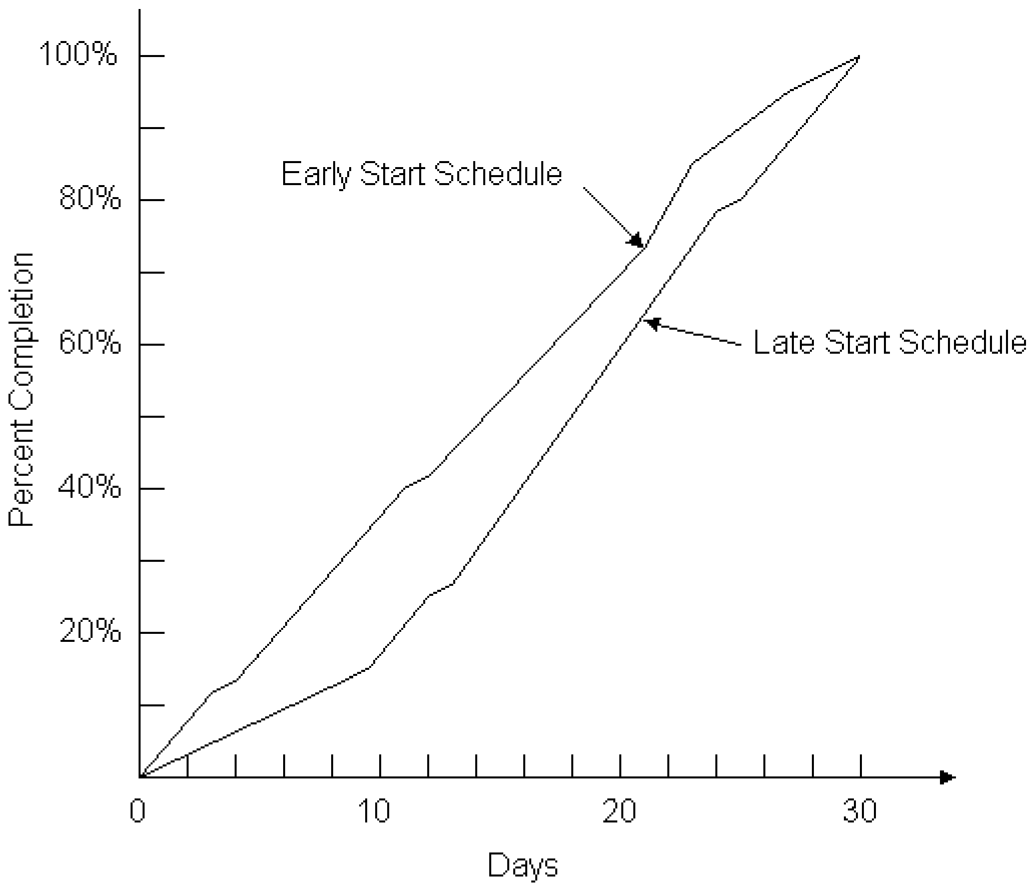


Figure 10-10 Example of Percentage Completion versus Time for Alternative Schedules with a Nine Activity Project

Time versus completion curves are also useful in project monitoring. Not only the history of the project can be indicated, but the future possibilities for earliest and latest start times. For example, Figure 10-11 illustrates a project that is forty percent complete after eight days for the nine activity example. In this case, the project is well ahead of the original schedule; some activities were completed in less than their expected durations. The possible earliest and latest start time schedules from the current project status are also shown on the figure.

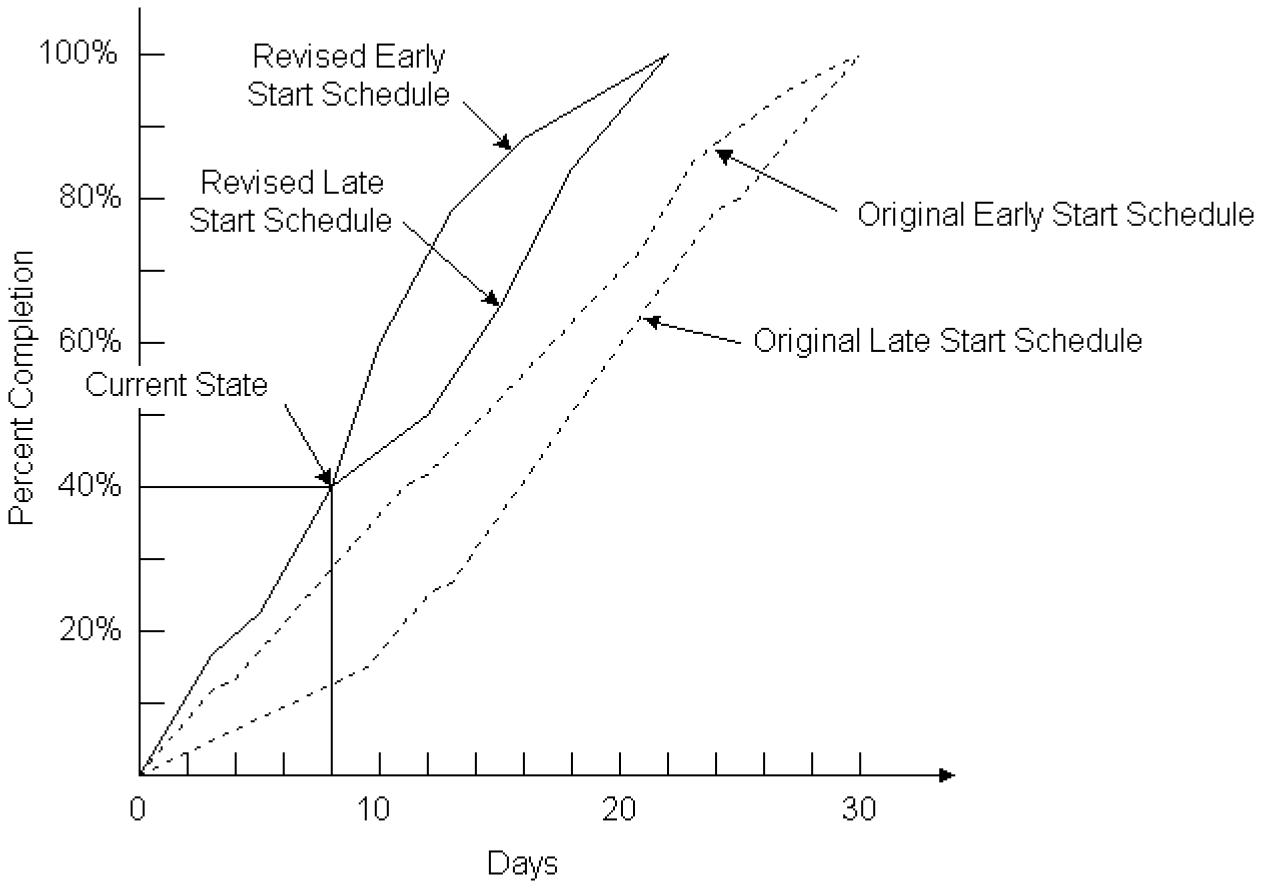


Figure 10-11 Illustration of Actual Percentage Completion versus Time for a Nine Activity Project Underway

Graphs of resource use over time are also of interest to project planners and managers. An example of resource use is shown in Figure 10-12 for the resource of total employment on the site of a project. This graph is prepared by summing the resource requirements for each activity at each time period for a particular project schedule. With limited resources of some kind, graphs of this type can indicate when the competition for a resource is too large to accommodate; in cases of this kind, resource constrained scheduling may be necessary as described in Section 10.9. Even without fixed resource constraints, a scheduler tries to avoid extreme fluctuations in the demand for labor or other resources since these fluctuations typically incur high costs for training, hiring, transportation, and management. Thus, a planner might alter a schedule through the use of available activity floats so as to level or smooth out the demand for resources. Resource graphs such as Figure 10-12 provide an invaluable indication of the potential trouble spots and the success that a scheduler has in avoiding them.

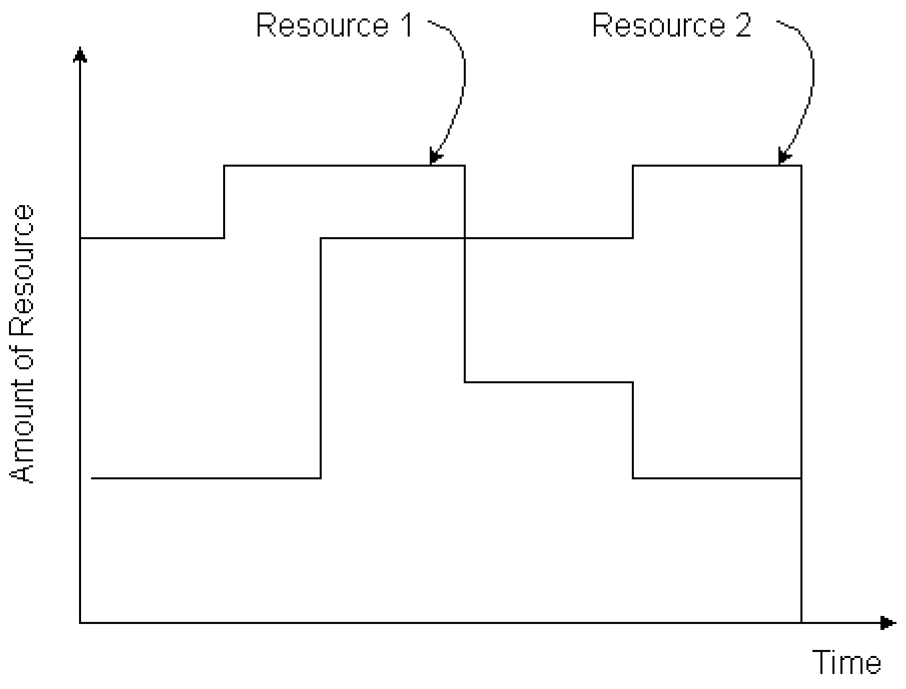


Figure 10-12 Illustration of Resource Use over Time for a Nine Activity Project

A common difficulty with project network diagrams is that too much information is available for easy presentation in a network. In a project with, say, five hundred activities, drawing activities so that they can be seen without a microscope requires a considerable expanse of paper. A large project might require the wall space in a room to include the entire diagram. On a computer display, a typical restriction is that less than twenty activities can be successfully displayed at the same time. The problem of displaying numerous activities becomes particularly acute when accessory information such as activity identifying numbers or phrases, durations and resources are added to the diagram.

One practical solution to this representation problem is to define sets of activities that can be represented together as a single activity. That is, for display purposes, network diagrams can be produced in which one "activity" would represent a number of real sub-activities. For example, an activity such as "foundation design" might be inserted in summary diagrams. In the actual project plan, this one activity could be sub-divided into numerous tasks with their own precedences, durations and other attributes. These sub-groups are sometimes termed *fragnets* for fragments of the full network. The result of this organization is the possibility of producing diagrams that summarize the entire project as well as detailed representations of particular sets of activities. The hierarchy of diagrams can also be introduced to the production of reports so that summary reports for groups of activities can be produced. Thus, detailed representations of particular activities such as plumbing might be prepared with all other activities either omitted or summarized in larger, aggregate activity representations. The CSI/MASTERSPEC activity definition codes described in Chapter 9 provide a widely adopted example of a hierarchical organization of this type. Even if summary reports and diagrams are prepared, the actual scheduling would use detailed activity characteristics, of course.

An example figure of a sub-network appears in Figure 10-13. Summary displays would include only a single node A to represent the set of activities in the sub-network. Note that precedence relationships shown in the master network would have to be interpreted with care since a particular precedence might be due to an activity that would not commence at the start of activity on the sub-network.

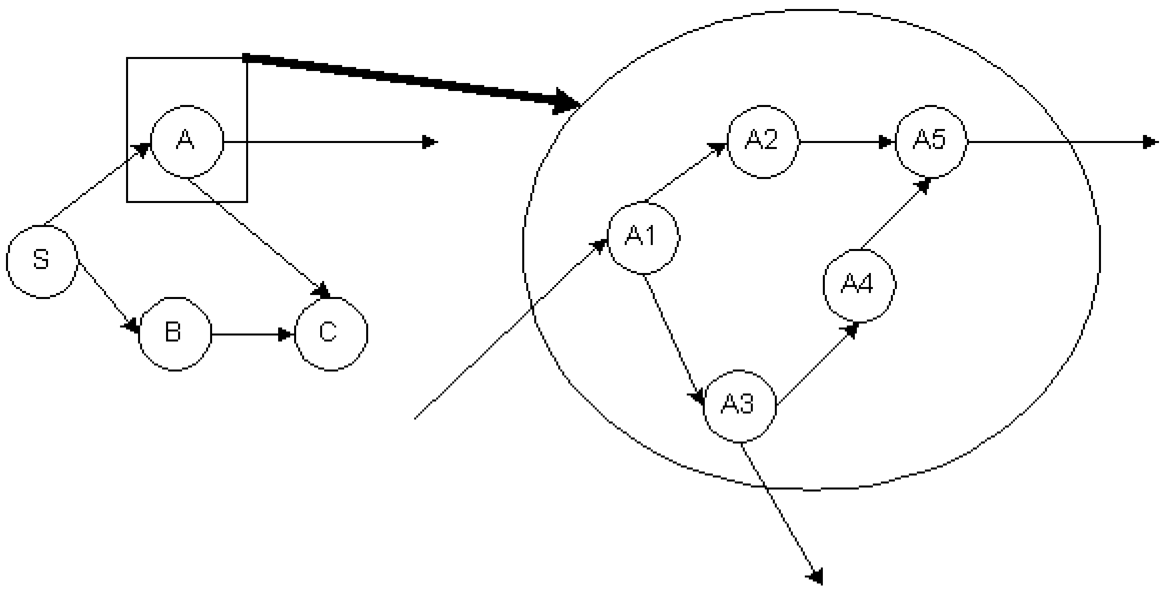


Figure 10-13 Illustration of a Sub-Network in a Summary Diagram

The use of graphical project representations is an important and extremely useful aid to planners and managers. Of course, detailed numerical reports may also be required to check the peculiarities of particular activities. But graphs and diagrams provide an invaluable means of rapidly communicating or understanding a project schedule. With computer based storage of basic project data, graphical output is readily obtainable and should be used whenever possible.

Finally, the scheduling procedure described in Section 10.3 simply counted days from the initial starting point. Practical scheduling programs include a calendar conversion to provide calendar dates for scheduled work as well as the number of days from the initiation of the project. This conversion can be accomplished by establishing a one-to-one correspondence between project dates and calendar dates. For example, project day 2 would be May 4 if the project began at time 0 on May 2 and no holidays intervened. In this calendar conversion, weekends and holidays would be excluded from consideration for scheduling, although the planner might overrule this feature. Also, the number of work shifts or working hours in each day could be defined, to provide consistency with the time units used is estimating activity durations. Project reports and graphs would typically use actual calendar days.

Back to top

10.6 Critical Path Scheduling for Activity-on-Node and with Leads, Lags, and Windows

Performing the critical path scheduling algorithm for activity-on-node representations is only a small variation from the activity-on-branch algorithm presented above. An example of the activity-on-node diagram for a seven activity network is shown in Figure 10-3. Some addition terminology is needed to account for the time delay at a node associated with the task activity. Accordingly, we define: ES(i) as the earliest start time for activity (and node) i, EF(i) is the earliest finish time for activity (and node) i, LS(i) is the latest start and LF(i) is the latest finish time for activity (and node) i. Table 10-7 shows the relevant calculations for the node numbering algorithm, the forward pass and the backward pass calculations.

TABLE 10-7 Critical Path Scheduling Algorithms (Activity-on-Node Representation)

Activity Numbering Algorithm
Step 1: Give the starting activity number 0. *Step 2*: Give the next number to any unnumbered activity whose predecessor activities are each already numbered. Repeat Step 2 until all activities are numbered.
Forward Pass
Step 1: Let E(0) = 0. *Step 2*: For j = 1,2,3,...,n (where n is the last activity), let ES(j) = maximum {EF(i)} where the maximum is computed over all activities (i) that have j as their successor. *Step 3*: EF(j) = ES(j) + D_j
Backward Pass
Step 1: Let L(n) equal the required completion time of the project. Note: L(n) must equal or exceed E(n). *Step 2*: For i = n-1, n-2, ..., 0, let LF(i) = minimum {LS(j)} where the minimum is computed over all activities (j) that have i as their predecessor. *Step 3*: LS(i) = LF(i) - D_i

For manual application of the critical path algorithm shown in Table 10-7, it is helpful to draw a square of four entries, representing the ES(i), EF(i), LS(i) and LF (i) as shown in Figure 10-14. During the forward pass, the boxes for ES(i) and EF(i) are filled in. As an exercise for the reader, the seven activity network in Figure 10-3 can be scheduled. Results should be identical to those obtained for the activity-on-branch calculations.

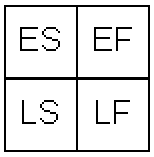


Figure 10-14 ES, EF, LS and LF Display for Hand Calculation of Critical Path for Activity-on-Node Representation

Building on the critical path scheduling calculations described in the previous sections, some additional capabilities are useful. Desirable extensions include the definition of allowable *windows* for activities and the introduction of more complicated precedence relationships among activities. For example, a planner may wish to have an activity of removing formwork from a new building component *follow* the concrete pour by some pre-defined lag period to allow setting. This delay would represent a required gap between the completion of a preceding activity and the start of a successor. The scheduling calculations to accommodate these complications will be described in this section. Again, the standard critical path scheduling assumptions of fixed activity durations and unlimited resource availability will be made here, although these assumptions will be relaxed in later sections.

A capability of many scheduling programs is to incorporate types of activity interactions in addition to the straightforward predecessor finish to successor start constraint used in Section 10.3. Incorporation of additional categories of interactions is often called *precedence diagramming*. [2] For example, it may be the case that installing concrete forms in a foundation trench might begin a few hours after the start of the trench excavation. This would be an example of a start-to-start constraint with a lead: the start of the trench-excavation activity would lead the start of the concrete-form-placement activity by a few hours. Eight separate categories of precedence constraints can be defined, representing greater than (leads) or less than (lags) time constraints for each of four different inter-activity relationships. These relationships are summarized in Table 10-8. Typical precedence relationships would be:

- **Direct or finish-to-start leads**
  The successor activity cannot start until the preceding activity is complete by at least the prescribed lead time (FS). Thus, the start of a successor activity must exceed the finish of the preceding activity by at least FS.
- **Start-to-start leads**
  The successor activity cannot start until work on the preceding activity has been underway by at least the prescribed lead time (SS).
- **Finish-to-finish leadss**
  The successor activity must have at least FF periods of work remaining at the completion of the preceding activity.
- **Start-to-finish leads**
  The successor activity must have at least SF periods of work remaining at the start of the preceding activity.

While the eight precedence relationships in Table 10-8 are all possible, the most common precedence relationship is the straightforward direct precedence between the finish of a preceding activity and the start of the successor activity with no required gap (so FS = 0).

TABLE 10-8 Eight Possible Activity Precedence Relationships

Relationship	Explanation
Finish-to-start Lead	Latest Finish of Predecessor $\geq$ Earliest Start of Successor + FS
Finish-to-start Lag	Latest Finish of Predecessor $\leq$ Earliest Start of Successor + FS
Start-to-start Lead	Earliest Start of Predecessor $\geq$ Earliest Start of Successor + SS
Start-to-start Lag	Earliest Start of Predecessor $\leq$ Earliest Start of Successor + SS
Finish-to-finish Lead	Latest Finish of Predecessor $\geq$ Earliest Finish of Successor + FF
Finish-to-finish Lag	Latest Finish of Predecessor $\leq$ Earliest Finish of Successor + FF
Start-to-finish Lead	Earliest Start of Predecessor $\geq$ Earliest Finish of Successor + SF
Start-to-finish Lag	Earliest Start of Predecessor $\leq$ Earliest Finish of Successor + SF

The computations with these lead and lag constraints are somewhat more complicated variations on the basic calculations defined in Table 10-1 for critical path scheduling. For example, a start-to-start lead would modify the calculation of the earliest start time to consider whether or not the necessary lead constraint was met:

(10.12) $$E(i) = max\left\{E(i) + D_{ij};\ E(i) + SS_{ij}\right\}$$

where SS_{ij} represents a start-to-start lead between activity (i,j) and any of the activities starting at event j.

The possibility of interrupting or *splitting* activities into two work segments can be particularly important to insure feasible schedules in the case of numerous lead or lag constraints. With activity splitting, an activity is divided into two sub-activities with a possible gap or idle time between work on the two subactivities. The computations for scheduling treat each sub-activity separately after a split is made. Splitting is performed to reflect available scheduling flexibility or to allow the development of a feasible schedule. For example, splitting may permit scheduling the early finish of a successor activity at a date *later* than the earliest start of the successor plus its duration. In effect, the successor activity is split into two segments with the later segment scheduled to finish after a particular time. Most commonly, this occurs when a constraint involving the finish time of two activities determines the required finish time of the successor. When this situation occurs, it is advantageous to split the successor activity into two so the first part of the successor activity can start earlier but still finish in accordance with the applicable finish-to-finish constraint.

Finally, the definition of activity *windows* can be extremely useful. An activity window defines a permissible period in which a particularly activity may be scheduled. To impose a window constraint, a planner could specify an earliest possible start time for an activity (WES) or a latest possible completion time (WLF). Latest possible starts (WLS) and earliest possible finishes (WEF) might also be imposed. In the extreme, a required start time might be insured by setting the earliest and latest window start times equal (WES = WLS). These window constraints would be in addition to the time constraints imposed by precedence relationships among the various project activities. Window constraints are particularly useful in enforcing milestone completion requirements on project activities. For example, a milestone activity may be defined with no duration but a latest possible completion time. Any activities preceding this milestone activity cannot be scheduled for completion after the milestone date. Window constraints are actually a special case of the other precedence constraints summarized above: windows are constraints in which the precedecessor activity is the project start. Thus, an earliest possible start time window (WES) is a start-to-start lead.

One related issue is the selection of an appropriate network representation. Generally, the activity-on-branch representation will lead to a more compact diagram and is also consistent with other engineering network representations of structures or circuits. [3] For example, the nine activities shown in Figure 10-4 result in an activity-on-branch network with six nodes and nine branches. In contrast, the comparable activity-on-node network shown in Figure 9-6 has eleven nodes (with the addition of a node for project start and completion) and fifteen branches. The activity-on-node diagram is more complicated and more difficult to draw, particularly since branches must be drawn crossing one another. Despite this larger size, an important practical reason to select activity-on-node diagrams is that numerous types of precedence relationships are easier to represent in these diagrams. For example, different symbols might be used on each of the branches in Figure 9-6 to represent direct precedences, start-to-start precedences, start-to-finish precedences, etc. Alternatively, the beginning and end points of the precedence links can indicate the type of lead or lag precedence relationship. Another advantage of activity-on-node representations is that the introduction of dummy links as in Figure 10-1 is not required. Either representation can be used for the critical path scheduling computations described earlier. In the absence of lead and lag precedence relationships, it is more common to select the compact activity-on-branch diagram, although a unified model for this purpose is described in Chapter 11. Of course, one reason to pick activity-on-branch or activity-on-node representations is that particular computer scheduling programs available at a site are based on one representation or the other. Since both representations are in common use, project managers should be familiar with either network representation.

Many commercially available computer scheduling programs include the necessary computational procedures to incorporate windows and many of the various precedence relationships described above. Indeed, the term "precedence diagramming" and the calculations associated with these lags seems to have first appeared in the user's manual for a computer scheduling program. [4]

If the construction plan suggests that such complicated lags are important, then these scheduling algorithms should be adopted. In the next section, the various computations associated with critical path scheduling with several types of leads, lags and windows are presented.

Back to top

10.7 Calculations for Scheduling with Leads, Lags and Windows

Table 10-9 contains an algorithmic description of the calculations required for critical path scheduling with leads, lags and windows. This description assumes an *activity-on-node* project network representation, since this representation is much easier to use with complicated precedence relationships. The possible precedence relationships accomadated by the procedure contained in Table 10-9 are finish-to-start leads, start-to-start leads, finish-to-finish lags and start-to-finish lags. Windows for earliest starts or latest finishes are also accomodated. Incorporating other precedence and window types in a scheduling procedure is also possible as described in Chapter 11. With an activity-on-node representation, we assume that an initiation and a termination activity are included to mark the beginning and end of the project. The set of procedures described in Table 10-9 does not provide for automatic splitting of activities.

TABLE 10-9 Critical Path Scheduling Algorithms with Leads, Lags and Windows (Activity-on-Node Representations)

Activity Numbering Algorithm
Step 1: Give the starting activity number 0. *Step 2*: Give the next number to any unnumbered activity whose predecessor activities are each already numbered. Repeat Step 2 until all activities are numbered.
Forward Pass Computations
Step 0: Set the earliest start and the earliest finish of the initial activity to zero: (ES(0) = EF(0) = 0). Repeat the following steps for each activity k = 0,1,2,...,m: *Step 1*: Compute the earliest start time (ES(k)) of activity k: ES(k) = Maximum {0; WES(k) for the earliest start window time, WEF(k) - D(k) for the earliest finish window time; EF(i) + FS(i,k) for each preceding activity with a F-S constraint; ES(i) + SS(i,k) for each preceding activity with a S-S constraint; EF(i) + FF(i,k) - D(k) for each preceding activity with a F-F constraint; ES(i) + SF(i,k) - D(k) for each preceding activity with a S-F constraint.} *Step* 2: Compute the earliest finish time EF(k) of activity k: EF(k) = ES(k) + D(k).
Backward Pass Computations
Step 0: Set the latest finish and latest start of the terminal activity to the early start time: LF(m) = LS(m) = ES(m) = EF(m) Repeat the following steps for each activity in reverse order, k = m-1,m-2,...,2,1,0: *Step 1*: Compute the latest finish time for activity k: LF(k) = Min{ LF(m), WLF(k) for the latest finish window time; WLS(k) + D(k) for the latest start window time; LS(j) - FS(k,j) for each succeeding activity with a F-S constraint; LF(j) - FF(k,j) for each succeeding activity with a FF constraint; LS(j) - SS(k,j) + D(k) for each succeeding activity with a SS constraint; LF(j) - SF(k,j) + D(k) for each succeeding activity with a SF constraint.} *Step* 2: Compute the latest start time for activity k:

$LS(k) = LF(k) - D(k)$

The first step in the scheduling algorithm is to sort activities such that no higher numbered activity precedes a lower numbered activity. With numbered activities, durations can be denoted D(k), where k is the number of an activity. Other activity information can also be referenced by the activity number. Note that node events used in *activity-on-branch* representations are not required in this case.

The forward pass calculations compute an earliest start time (ES(k)) and an earliest finish time (EF(k)) for each activity in turn (Table 10-9). In computing the earliest start time of an activity k, the earliest start window time (WES), the earliest finish window time (WEF), and each of the various precedence relationships must be considered. Constraints on finish times are included by identifying minimum finish times and then subtracting the activity duration. A default earliest start time of day 0 is also insured for all activities. A second step in the procedure is to identify each activity's earliest finish time (EF(k)).

The backward pass calculations proceed in a manner very similar to those of the forward pass (Table 10-9). In the backward pass, the latest finish and the latest start times for each activity are calculated. In computing the latest finish time, the latest start time is identified which is consistent with precedence constraints on an activity's starting time. This computation requires a minimization over applicable window times and all successor activities. A check for a feasible activity schedule can also be imposed at this point: if the late start time is less than the early start time ($LS(k) < ES(k)$), then the activity schedule is not possible.

The result of the forward and backward pass calculations are the earliest start time, the latest start time, the earliest finish time, and the latest finish time for each activity. The activity float is computed as the latest start time less the earliest start time. Note that window constraints may be instrumental in setting the amount of float, so that activities without any float may either lie on the critical path or be constrained by an allowable window.

To consider the possibility of activity splitting, the various formulas for the forward and backward passes in Table 10-9 must be modified. For example, in considering the possibility of activity splitting due to start-to-start lead (SS), it is important to ensure that the preceding activity has been underway for at least the required lead period. If the preceding activity was split and the first sub-activity was not underway for a sufficiently long period, then the following activity cannot start until the first plus the second sub-activities have been underway for a period equal to SS(i,k). Thus, in setting the earliest start time for an activity, the calculation takes into account the duration of the first subactivity (DA(i)) for preceding activities involving a start-to-start lead. Algebraically, the term in the earliest start time calculation pertaining to start-to-start precedence constraints (ES(i) + SS(i,k)) has two parts with the possibility of activity splitting:

(10.13) $ES(i) + SS(i,k)$

(10.14) $EF(i) - D(i) + SS(i,k)$ for split preceding activities with $DA(i) < SS(i,k)$

where DA(i) is the duration of the first sub-activity of the preceding activity.

The computation of earliest finish time involves similar considerations, except that the finish-to-finish and start-to-finish lag constraints are involved. In this case, a maximization over the following terms is required:

(10.15)
EF(k) = Maximum{ES(k) + D(k),
EF(i) + FF(i,k) for each preceding activity with a FF precedence,
ES(i) + SF(i,k) for each preceding activity with a SF precedence and which is not split,
EF(i) - D(i) + SF(i,k) for each preceding activity with a SF precedence and which is split}

Finally, the necessity to split an activity is also considered. If the earliest possible finish time is greater than the earliest start time plus the activity duration, then the activity must be split.

Another possible extension of the scheduling computations in Table 10-9 would be to include a duration modification capability during the forward and backward passes. This capability would permit alternative work calendars for different activities or for modifications to reflect effects of time of the year on activity durations. For example, the duration of outside work during winter months would be increased. As another example, activities with weekend work permitted might have their weekday durations shortened to reflect weekend work accomplishments.

Example 10-4: Impacts of precedence relationships and windows

To illustrate the impacts of different precedence relationships, consider a project consisting of only two activities in addition to the start and finish. The start is numbered activity 0, the first activity is number 1, the second activity is number 2, and the finish is activity 3. Each activity is assumed to have a duration of five days. With a direct finish-to-start precedence relationship without a lag, the critical path calculations reveal:
ES(0) = 0
ES(1) = 0
EF(1) = ES(1) + D(1) = 0 + 5 = 5
ES(2) = EF(1) + FS(1,2) = 5 + 0 = 5
EF(2) = ES(2) + D(2) = 5 + 5 = 10
ES(3) = EF(2) + FS(2,3) = 10 + 0 = 10 = EF(3)

So the earliest project completion time is ten days.

With a start-to-start precedence constraint with a two day lead, the scheduling calculations are:

ES(0) = 0
ES(1) = 0
EF(1) = ES(1) + D(1) = 0 + 5 = 5
ES(2) = ES(1) + SS(1,2) = 0 + 2 = 2

EF(2) = ES(2) + D(2) = 2 + 5 = 7
ES(3) = EF(2) + FS(2,3) = 7 + 0 = 7.

In this case, activity 2 can begin two days after the start of activity 1 and proceed in parallel with activity 1. The result is that the project completion date drops from ten days to seven days.

Finally, suppose that a finish-to-finish precedence relationship exists between activity 1 and activity 2 with a two day lag. The scheduling calculations are:

ES(0) = 0 = EF(0)
ES(1) = EF(0) + FS(0,1) = 0 + 0 = 0
EF(1) = ES(1) + D(1) = 0 + 5 = 5
ES(2) = EF(1) + FF(1,2) - D(2) = 5 + 2 - 5 = 2
EF(2) = ES(2) + D(2) = 2 + 5 = 7
ES(3) = EF(2) + FS(2,3) = 7 + 0 = 7 = EF(3)

In this case, the earliest finish for activity 2 is on day seven to allow the necessary two day lag from the completion of activity 1. The minimum project completion time is again seven days.

Example 10-5: Scheduling in the presence of leads and windows.

As a second example of the scheduling computations involved in the presence of leads, lags and windows, we shall perform the calculations required for the project shown in Figure 10-15. Start and end activities are included in the project diagram, making a total of eleven activities. The various windows and durations for the activities are summarized in Table 10-10 and the precedence relationships appear in Table 10-11. Only earliest start (WES) and latest finish (WLF) window constraints are included in this example problem. All four types of precedence relationships are included in this project. Note that two activities may have more than one type of precedence relationship at the same time; in this case, activities 2 and 5 have both S-S and F-F precedences. In Figure 10-15, the different precedence relationships are shown by links connecting the activity nodes. The type of precedence relationship is indicated by the beginning or end point of each arrow. For example, start-to-start precedences go from the left portion of the preceding activity to the left portion of the following activity. Application of the activity sorting algorithm (Table 10-9) reveals that the existing activity numbers are appropriate for the critical path algorithm. These activity numbers will be used in the forward and backward pass calculations.

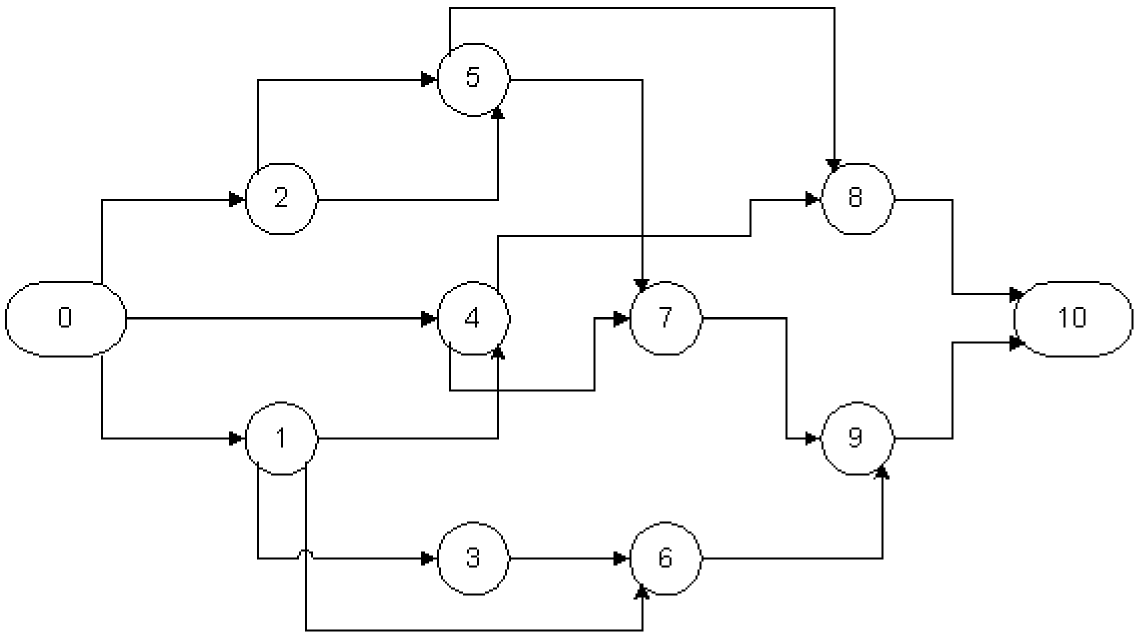


Figure 10-15 Example Project Network with Lead Precedences

TABLE 10-10 Predecessors, Successors, Windows and Durations for an Example Project

Activity Number	Predecessors	Successors	Earliest Start Window	Latest Finish Window	Activity Duration
0	---	1, 2, 4	---	---	0
1	0	3, 4, 6	---	---	2
2	0	5	---	---	5
3	1	6	2	---	4
4	0	7, 8	---	---	3
5	2, 2	7, 8	---	16	5
6	1, 3	9	6	16	6
7	4, 5	9	---	---	2
8	4, 5	10	---	---	4
9	6, 7	10	---	16	5
10	8, 9	---	---	---	0

TABLE 10-11 Precedences in a Eleven Activity Project Example

Predecessor	Successor	Type	Lead
0	1	FS	0
0	2	FS	0
0	4	FS	0
1	3	SS	1
1	4	SF	1

1	6	FS	2
2	5	SS	2
2	5	FF	2
3	6	FS	0
4	7	SS	2
4	8	FS	0
5	7	FS	1
5	8	SS	3
6	9	FF	4
7	9	FS	0
8	10	FS	0
9	10	FS	0

During the forward pass calculations (Table 10-9), the earliest start and earliest finish times are computed for each activity. The relevant calculations are:

ES(0) = EF(0) = 0
ES(1) = Max{0; EF(0) + FS(0,1)} = Max {0; 0 + 0} = 0.
EF(1) = ES(1) + D(1) = 0 + 2 = 2
ES(2) = Max{0; EF(0) + FS(0,1)} = Max{0; 0 + 0} = 0.
EF(2) = ES(2) + D(2) = 0 + 5 = 5
ES(3) = Max{0; WES(3); ES(1) + SS(1,3)} = Max{0; 2; 0 + 1} = 2.
EF(3) = ES(3) + D(3) = 2 + 4 = 6

Note that in the calculation of the earliest start for activity 3, the start was delayed to be consistent with the earliest start time window.

ES(4) = Max{0; ES(0) + FS(0,1); ES(1) + SF(1,4) - D(4)} = Max{0; 0 + 0; 0+1-3} = 0.
EF(4) = ES(4) + D(4) = 0 + 3 = 3
ES(5) = Max{0; ES(2) + SS(2,5); EF(2) + FF(2,5) - D(5)} = Max{0; 0+2; 5+2-5} = 2
EF(5) = ES(5) + D(5) = 2 + 5 = 7
ES(6) = Max{0; WES(6); EF(1) + FS(1,6); EF(3) + FS(3,6)} = Max{0; 6; 2+2; 6+0} = 6
EF(6) = ES(6) + D(6) = 6 + 6 = 12
ES(7) = Max{0; ES(4) + SS(4,7); EF(5) + FS(5,7)} = Max{0; 0+2; 7+1} = 8
EF(7) = ES(7) + D(7) = 8 + 2 = 10
ES(8) = Max{0; EF(4) + FS(4,8); ES(5) + SS(5,8)} = Max{0; 3+0; 2+3} = 5
EF(8) = ES(8) + D(8) = 5 + 4 = 9
ES(9) = Max{0; EF(7) + FS(7,9); EF(6) + FF(6,9) - D(9)} = Max{0; 10+0; 12+4-5} = 11
EF(9) = ES(9) + D(9) = 11 + 5 = 16
ES(10) = Max{0; EF(8) + FS(8,10); EF(9) + FS(9,10)} = Max{0; 9+0; 16+0} = 16
EF(10) = ES(10) + D(10) = 16

As the result of these computations, the earliest project completion time is found to be 16 days.

The backward pass computations result in the latest finish and latest start times for each activity. These calculations are:

LF(10) = LS(10) = ES(10) = EF(10) = 16
LF(9) = Min{WLF(9); LF(10);LS(10) - FS(9,10)} = Min{16;16; 16-0} = 16
LS(9) = LF(9) - D(9) = 16 - 5 = 11
LF(8) = Min{LF(10); LS(10) - FS(8,10)} = Min{16; 16-0} = 16
LS(8) = LF(8) - D(8) = 16 - 4 = 12
LF(7) = Min{LF(10); LS(9) - FS(7,9)} = Min{16; 11-0} = 11
LS(7) = LF(7) - D(7) = 11 - 2 = 9
LF(6) = Min{LF(10); WLF(6); LF(9) - FF(6,9)} = Min{16; 16; 16-4} = 12
LS(6) = LF(6) - D(6) = 12 - 6 = 6
LF(5) = Min{LF(10); WLF(10); LS(7) - FS(5,7); LS(8) - SS(5,8) + D(8)} = Min{16; 16; 9-1; 12-3+4} = 8
LS(5) = LF(5) - D(5) = 8 - 5 = 3
LF(4) = Min{LF(10); LS(8) - FS(4,8); LS(7) - SS(4,7) + D(7)} = Min{16; 12-0; 9-2+2} = 9
LS(4) = LF(4) - D(4) = 9 - 3 = 6
LF(3) = Min{LF(10); LS(6) - FS(3,6)} = Min{16; 6-0} = 6
LS(3) = LF(3) - D(3) = 6 - 4 = 2
LF(2) = Min{LF(10); LF(5) - FF(2,5); LS(5) - SS(2,5) + D(5)} = Min{16; 8-2; 3-2+5} = 6
LS(2) = LF(2) - D(2) = 6 - 5 = 1
LF(1) = Min{LF(10); LS(6) - FS(1,6); LS(3) - SS(1,3) + D(3); Lf(4) - SF(1,4) + D(4)}
LS(1) = LF(1) - D(1) = 2 -2 = 0
LF(0) = Min{LF(10); LS(1) - FS(0,1); LS(2) - FS(0,2); LS(4) - FS(0,4)} = Min{16; 0-0; 1-0; 6-0} = 0
LS(0) = LF(0) - D(0) = 0

The earliest and latest start times for each of the activities are summarized in Table 10-12. Activities without float are 0, 1, 6, 9 and 10. These activities also constitute the critical path in the project. Note that activities 6 and 9 are related by a finish-to-finish precedence with a 4 day lag. Decreasing this lag would result in a reduction in the overall project duration.

TABLE 10-12 Summary of Activity Start and Finish Times for an Example Problem

Activity	Earliest Start	Latest Start	Float
0	0	0	0
1	0	0	0
2	0	1	1
3	0	2	2
4	0	6	6
5	2	3	1
6	6	6	0
7	8	9	1
8	5	12	7
9	11	11	0
10	16	16	0

Back to top

10.8 Resource Oriented Scheduling

Resource constrained scheduling should be applied whenever there are limited resources available for a project and the competition for these resources among the project activities is keen. In effect, delays are liable to occur in such cases as activities must wait until common resources become available. To the extent that resources are limited and demand for the resource is high, this waiting may be considerable. In turn, the congestion associated with these waits represents increased costs, poor productivity and, in the end, project delays. Schedules made without consideration for such bottlenecks can be completely unrealistic.

Resource constrained scheduling is of particular importance in managing multiple projects with fixed resources of staff or equipment. For example, a design office has an identifiable staff which must be assigned to particular projects and design activities. When the workload is heavy, the designers may fall behind on completing their assignments. Government agencies are particularly prone to the problems of fixed staffing levels, although some flexibility in accomplishing tasks is possible through the mechanism of contracting work to outside firms. Construction activities are less susceptible to this type of problem since it is easier and less costly to hire additional personnel for the (relatively) short duration of a construction project. Overtime or double shift work also provide some flexibility.

Resource oriented scheduling also is appropriate in cases in which unique resources are to be used. For example, scheduling excavation operations when one only excavator is available is simply a process of assigning work tasks or job segments on a day by day basis while insuring that appropriate precedence relationships are maintained. Even with more than one resource, this manual assignment process may be quite adequate. However, a planner should be careful to insure that necessary precedences are maintained.

Resource constrained scheduling represents a considerable challenge and source of frustration to researchers in mathematics and operations research. While algorithms for optimal solution of the resource constrained problem exist, they are generally too computationally expensive to be practical for all but small networks (of less than about 100 nodes). [5] The difficulty of the resource constrained project scheduling problem arises from the combinatorial explosion of different resource assignments which can be made and the fact that the decision variables are integer values representing all-or-nothing assignments of a particular resource to a particular activity. In contrast, simple critical path scheduling deals with continuous time variables. Construction projects typically involve many activities, so optimal solution techniques for resource allocation are not practical.

One possible simplification of the resource oriented scheduling problem is to ignore precedence relationships. In some applications, it may be impossible or unnecessary to consider precedence constraints among activities. In these cases, the focus of scheduling is usually on efficient utilization of project resources. To insure minimum cost and delay, a project manager attempts to minimize the amount of time that resources are unused and to minimize the waiting time for scarce resources. This resource oriented scheduling is often formalized as a problem of "job shop" scheduling in which numerous tasks are to be scheduled for completion and a variety of discrete resources need to perform operations to complete the tasks. Reflecting the original orientation towards manufacturing applications, tasks are usually referred to as "jobs" and resources to be scheduled are designated "machines." In the provision of constructed facilities, an analogy would be an architectural/engineering design office in which numerous design related tasks are to be accomplished by individual professionals in different departments. The scheduling problem is to insure efficient use of the individual professionals (i.e. the resources) and to complete specific tasks in a timely manner.

The simplest form of resource oriented scheduling is a reservation system for particular resources. In this case, competing activities or users of a resource pre-arrange use of the resource for a particular time period. Since the resource assignment is known in advance, other users of the resource can schedule their activities more effectively. The result is less waiting or "queuing" for a resource. It is also possible to inaugurate a preference system within the reservation process so that high-priority activities can be accomadated directly.

In the more general case of multiple resources and specialized tasks, practical resource constrained scheduling procedures rely on heuristic procedures to develop good but not necessarily optimal schedules. While this is the occasion for considerable anguish among researchers, the heuristic methods will typically give fairly good results. An example heuristic method is provided in the next section. Manual methods in which a human scheduler revises a critical path schedule in light of resource constraints can also work relatively well. Given that much of the data and the network representation used in forming a project schedule are uncertain, the results of applying heuristic procedures may be quite adequate in practice.

Example 10-6: A Reservation System [6]

A recent construction project for a high-rise building complex in New York City was severely limited in the space available for staging materials for hauling up the building. On the four building site, thirty-eight separate cranes and elevators were available, but the number of movements of men, materials and equipment was expected to keep the equipment very busy. With numerous sub-contractors desiring the use of this equipment, the potential for delays and waiting in the limited staging area was considerable. By implementing a crane reservation system, these problems were nearly entirely avoided. The reservation system required contractors to telephone one or more days in advance to reserve time on a particular crane. Time were available on a first-come, first-served basis (i.e. first call, first choice of available slots). Penalties were imposed for making an unused reservation. The reservation system was also computerized to permit rapid modification and updating of information as well as the provision of standard reservation schedules to be distributed to all participants.

Example 10-7: Heuristic Resource Allocation

Suppose that a project manager has eleven pipe sections for which necessary support structures and materials are available in a particular week. To work on these eleven pipe sections, five crews are available. The allocation problem is to assign the crews to the eleven pipe sections. This allocation would consist of a list of pipe sections allocated to each crew for work plus a recommendation on the appropriate sequence to undertake the work. The project manager might make assignments to minimize completion time, to insure continuous work on the pipeline (so that one section on a pipeline run is not left incomplete), to reduce travel time between pipe sections, to avoid congestion among the different crews, and to balance the workload among the crews. Numerous trial solutions could be rapidly generated, especially with the aid of an electronic spreadsheet. For example, if the nine sections had estimated work durations for each of the fire crews as shown in Table 10-13, then the allocations shown in Figure 10-16 would result in a minimum completion time.

TABLE 10-13 Estimated Required Time for Each Work Task in a Resource Allocation Problem

Section	Work Duration
A	9
B	9
C	8
D	8
E	7
F	7
G	6
H	6
I	6
J	5
K	5

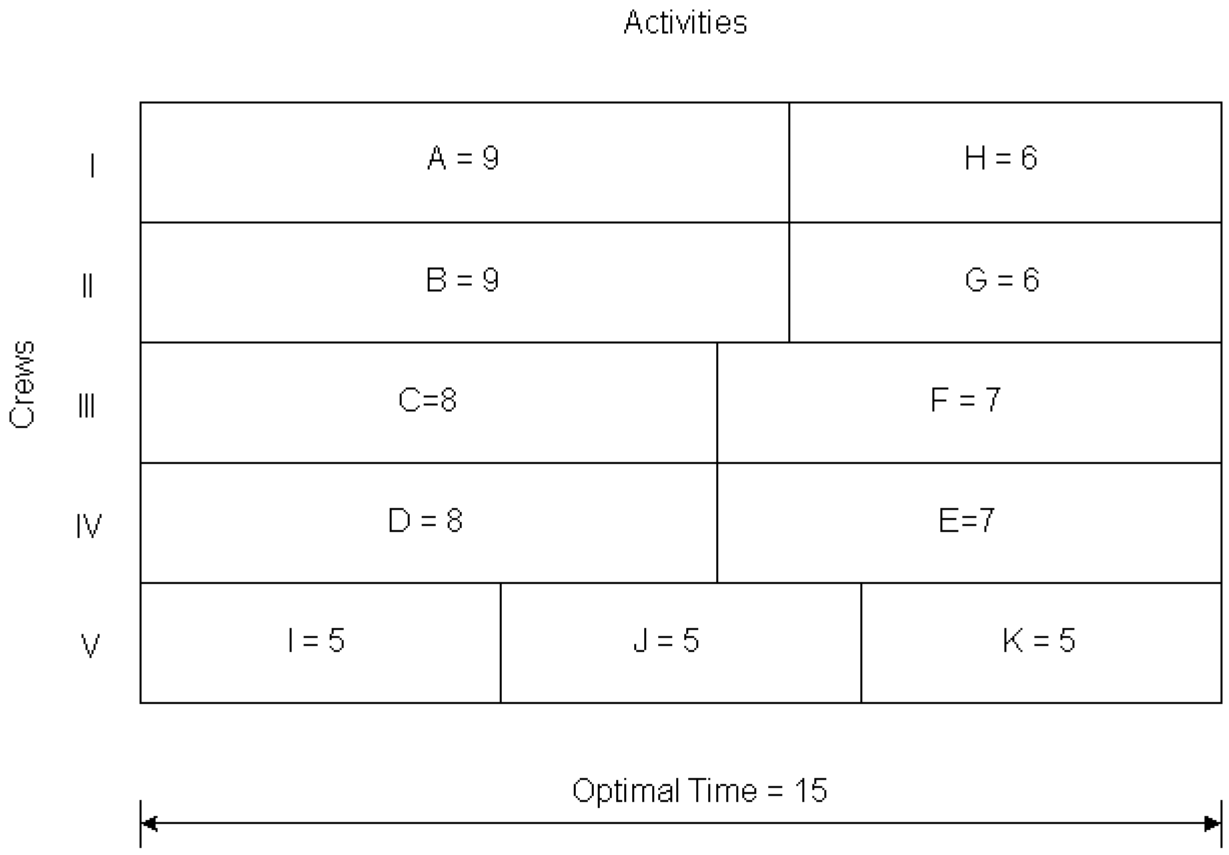


Figure 10-16 Example Allocation of Crews to Work Tasks

Example 10-8: Algorithms for Resource Allocation with Bottleneck Resources

In the previous example, suppose that a mathematical model and solution was desired. For this purpose, we define a binary (i.e. 0 or 1 valued) decision variable for each pipe section and crew, x_{ij}, where $x_{ij} = 1$ implies that section i was assigned to crew j and $x_{ij} = 0$ implied that section i was not assigned to crew j. The time required to complete each section is t_i. The overall time to complete the nine sections is denoted z. In this case, the problem of minimizing overall completion time is:

$$z = maximum\left(\sum_{i=1}^{11} t_i x_{i1};\ \sum_{i=1}^{11} t_i x_{i2};\ \sum_{i=1}^{11} t_i x_{i3};\ \sum_{i=1}^{11} t_i x_{i4};\ \sum_{i=}^{11}\right.$$

subject to the constraints:

$$\sum_{j=1}^{5} x_{ij} = 1$$

for each section i

x_{ij} is 0 or 1

where the constraints simply insure that each section is assigned to one and only one crew. A modification permits a more conventional mathematical formulation, resulting in a generalized bottleneck assignment problem:

Minimize z
subject to the constraints:

$$z \geq \sum_{i=1}^{11} t_i x_{ij}$$

for each crew j

$$\sum_{j=1}^{5} x_{ij} = 1$$

for each section i

x_{ij} is 0 or 1

This problem can be solved as an integer programming problem, although at considerable computational expense. A common extension to this problem would occur with differential productivities for each crew, so that the time to complete an activity, t_{ij}, would be defined for each crew. Another modification to this problem would substitute a cost factor, c_j, for the time factor, t_j, and attempt to minimize overall costs rather than completion time.

Back to top

10.9 Scheduling with Resource Constraints and Precedences

The previous section outlined resource oriented approaches to the scheduling problem. In this section, we shall review some general approaches to integrating both concerns in scheduling.

Two problems arise in developing a resource constrained project schedule. First, it is not necessarily the case that a critical path schedule is feasible. Because one or more resources might be needed by numerous activities, it can easily be the case that the shortest project duration identified by the critical path scheduling calculation is impossible. The difficulty arises because critical path scheduling assumes that no resource availability problems or bottlenecks will arise. Finding a feasible or possible schedule is the first problem in resource constrained scheduling. Of course, there may be a numerous possible schedules which conform with time and resource constraints. As a second problem, it is also desirable to determine schedules which have low costs or, ideally, the lowest cost.

Numerous heuristic methods have been suggested for resource constrained scheduling. Many begin from critical path schedules which are modified in light of the resource constraints. Others begin in the opposite fashion by introducing resource constraints and then imposing precedence constraints on the activities. Still others begin with a ranking or classification of activities into priority groups for special attention in scheduling. [7] One type of heuristic may be better than another for different types of problems. Certainly, projects in which only an occasional resource constraint exists might be best scheduled starting from a critical path schedule. At the other extreme, projects with numerous important resource constraints might be best scheduled by considering critical resources first. A mixed approach would be to proceed simultaneously considering precedence and resource constraints.

A simple modification to critical path scheduling has been shown to be effective for a number of scheduling problems and is simple to implement. For this heuristic procedure, critical path scheduling is applied initially. The result is the familiar set of possible early and late start times for each activity. Scheduling each activity to begin at its earliest possible start time may result in more than one activity requiring a particular resource at the same time. Hence, the initial schedule may not be feasible. The heuristic proceeds by identifying cases in which activities compete for a resource and selecting one activity to proceed. The start time of other activities are then shifted later in time. A simple rule for choosing which activity has priority is to select the activity with the earliest CPM late start time (calculated as $LS(i,j) = L(j)-D_{ij}$) among those activities which are both feasible (in that all their precedence requirements are satisfied) and competing for the resource. This decision rule is applied from the start of the project until the end for each type of resource in turn.

The order in which resources are considered in this scheduling process may influence the ultimate schedule. A good heuristic to employ in deciding the order in which resources are to be considered is to consider more important resources first. More important resources are those that have high costs or that are likely to represent an important bottleneck for project completion. Once important resources are scheduled, other resource allocations tend to be much easier. The resulting scheduling procedure is described in Table 10-14.

The late start time heuristic described in Table 10-14 is only one of many possible scheduling rules. It has the advantage of giving priority to activities which must start sooner to finish the project on time. However, it is *myopic* in that it doesn't consider trade-offs among resource types nor the changes in the late start time that will be occurring as activities are shifted later in time. More complicated rules can be devised to incorporate broader knowledge of the project schedule. These complicated rules require greater computational effort and may or may not result in scheduling improvements in the end.

TABLE 10-14 A Resource-Oriented Scheduling Procedure

Step 1:

Rank all resources from the most important to the least important, and number the resources i = 1,2,3,...,m.

Step 2:

Set the scheduled start time for each activity to the earliest start time.

For each resource i = 1,2,3,...,m in turn:

Step 3:

Start at the project beginning, so set t = 0.

Step 4:

Compute the demand for resource i at time t by summing up the requirements for resource i for all activities scheduled to be underway at time t.

If demand for resource i in time t is greater than the resource availability, then select the activity with the greatest late start time requiring resource i at time t, and shift its scheduled start time to time t+1.

Repeat Step 4 until the resource constraint at time t for resource i is satisfied.

Step 5:

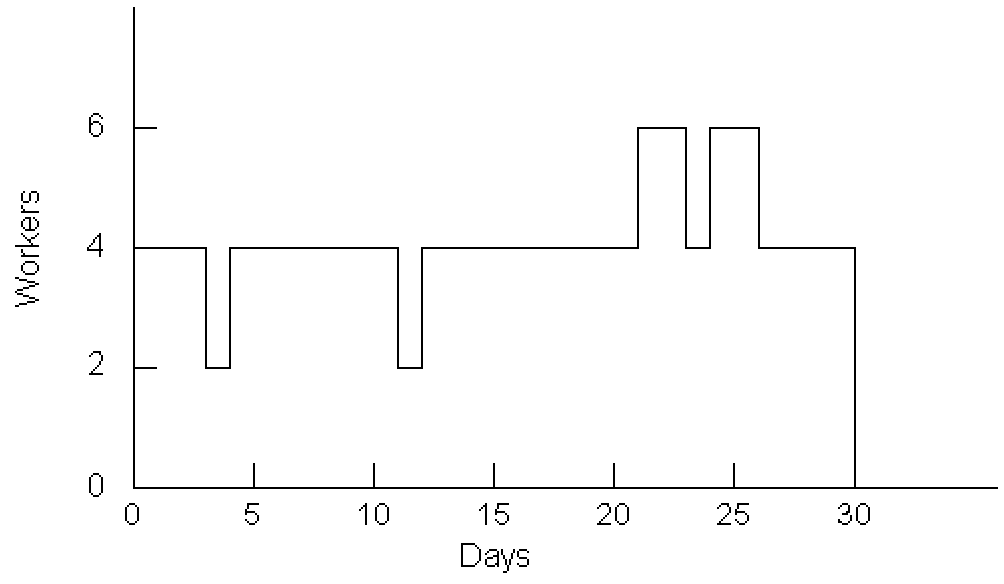
Repeat step 4 for each project period in turn, setting t = t+1.

Example 10-9: Resource constrained scheduling with nine activities.

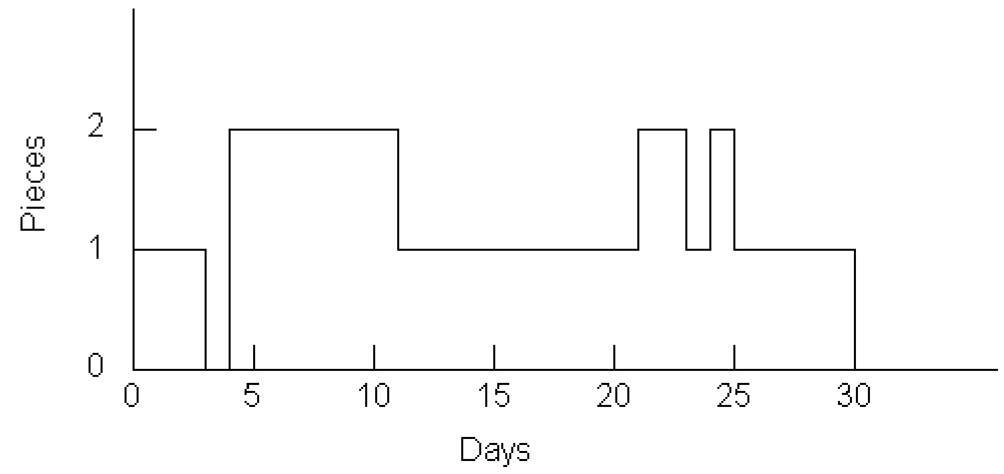
As an example of resource constrained scheduling, we shall re-examine the nine activity project discussed in Section 10.3. To begin with, suppose that four workers and two pieces of equipment such as backhoes are available for the project. The required resources for each of the nine project activities are summarized in Table 10-15. Graphs of resource requirements over the 30 day project duration are shown in Figure 10-17. Equipment availability in this schedule is not a problem. However, on two occasions, more than the four available workers are scheduled for work. Thus, the existing project schedule is infeasible and should be altered.

TABLE 10-15 Resources Required and Starting Times for a Nine Activity Project

Activity	Workers Required	Equipment Required	Earliest Start Time	Latest Start Time	Duration
A	2	0	0	0	4
B	2	1	0	9	3
C	2	1	4	4	8
D	2	1	4	15	7
E	2	1	12	13	9
F	2	0	12	12	12
G	2	1	21	22	2
H	2	1	21	25	5
I	4	1	24	24	6



(a) Workers Required



(b) Equipment Required

Figure 10-17 Resources Required over Time for Nine Activity Project: Schedule I

The first resource problem occurs on day 21 when activity F is underway and activities G and H are scheduled to start. Applying the latest start time heuristic to decide which activity should start, the manager should re-schedule activity H since it has a later value of LS(i,j), i.e., day 25 versus day 22 as seen in Table 10-15. Two workers become available on day 23 after the completion of activity G. Since activity H is the only activity which is feasible at that time, it is scheduled to begin. Two workers also become available on day 24 at the completion of activity F. At this point, activity I is available for starting. If possible, it would be scheduled to begin with only two workers until the completion of activity H on day 28. If all 4 workers were definitely required, then activity I would be scheduled to begin on day 28. In this latter case, the project duration would be 34 days, representing a 4 day increase due to the limited number of workers available.

Example 10-10: Additional resource constraints.

As another example, suppose that only one piece of equipment was available for the project. As seen in Figure 10-17, the original schedule would have to be significantly modified in this case. Application of the resource constrained scheduling heuristic proceeds as follows as applied to the original project schedule:

1. On day 4, activities D and C are both scheduled to begin. Since activity D has a larger value of late start time, it should be re-scheduled.
2. On day 12, activities D and E are available for starting. Again based on a later value of late start time (15 versus 13), activity D is deferred.
3. On day 21, activity E is completed. At this point, activity D is the only feasible activity and it is scheduled for starting.
4. On day 28, the planner can start either activity G or activity H. Based on the later start time heuristic, activity G is chosen to start.
5. On completion of activity G at day 30, activity H is scheduled to begin.

The resulting profile of resource use is shown in Figure 10-18. Note that activities F and I were not considered in applying the heuristic since these activities did not require the special equipment being considered. In the figure, activity I is scheduled after the completion of activity H due to the requirement of 4 workers for this activity. As a result, the project duration has increased to 41 days. During much of this time, all four workers are not assigned to an activity. At this point, a prudent planner would consider whether or not it would be cost effective to obtain an additional piece of equipment for the project.

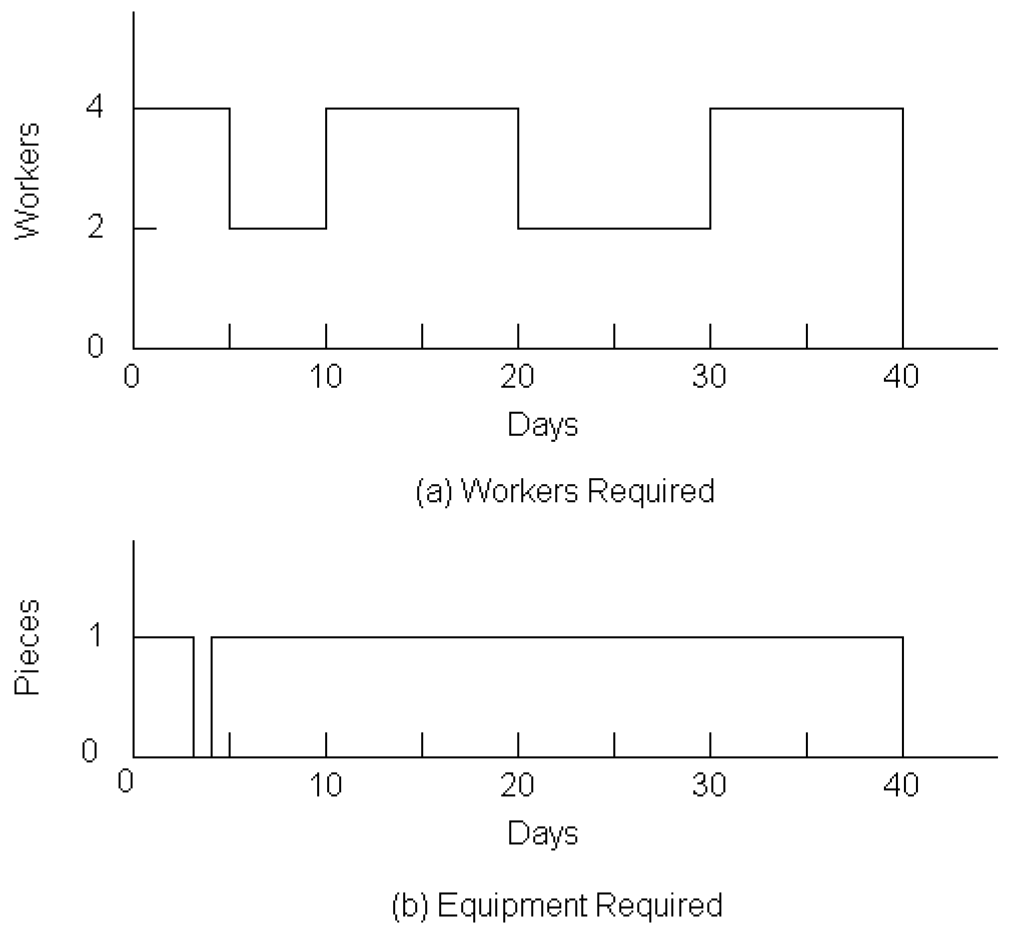


Figure 10-18 Resources Required over Time for Nine Activity Project: Schedule II

Back to top

10.10 References

1. Au, T. *Introduction to Systems Engineering--Deterministic Models*, Addison-Wesley, Reading, MA, 1973, Chapter 8.
2. Baker, K., *An Introduction to Sequencing and Scheduling*, John Wiley, 1974.
3. Jackson, M.J., *Computers in Construction Planning and Control*, Allen & Unwin, 1986.
4. Moder, J., C. Phillips and E. Davis, *Project Management with CPM, PERT and Precedence Diagramming*, Van Nostrand Reinhold Company, Third Edition,1983.
5. Willis, E. M., *Scheduling Construction Projects*, John Wiley & Sons, 1986.

Back to top

10.11 Problems

1 to 4.
Construct an activity-on-branch network from the precedence relationships of activities in the project given in the table for the problem, Tables 10-16 to 10-19.

TABLE 10-16

Activity	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Predecessors	---	A	A	---	B	C,D	C,D	D	H	F	E,J	F	G,I	G,I	L,N
Duration	6	7	1	14	5	8	9	3	5	3	4	12	6	2	7

TABLE 10-17

Activity	A	B	C	D	E	F	G	H	I	J	K	L	M	N
Predecessors	---	A	B	C	D,G	A	F,J	---	H	I	F,J	H	L	K,M
Duration	5	6	3	4	5	8	3	3	2	7	2	7	4	3

TABLE 10-18

Activity	A	B	C	D	E	F	G	H	I	J	K	L
Predecessors	---	---	---	A	B	C	B,D	C,E	F	F	E,G,I	H,J
Duration	6	12	16	5	3	10	9	4	5	3	10	6

TABLE 10-19

Activity	A	B	C	D	E	F	G	H	I	J	K	L	M
Predecessors	---	---	---	C	C	B,E	A,F	B,E	B,E	B,E	D,J	G,H	I,K,L
Duration	3	6	2	3	8	5	7	10	6	6	8	3	4

5 to 8.
Determine the critical path and all slacks for the projects in Tables 10-16 to 10-19.

9. Suppose that the precedence relationships for Problem 1 in Table 10-16 are all direct finish-to-start relationships with no lags except for the following:
   - B to E: S-S with a lag of 2.
   - D to H: F-F with a lag of 3.
   - F to L: S-S with a lag of 2.
   - G to N: S-S with a lag of 1.
   - G to M: S-S with a lag of 2.

   Formulate an activity-on-node network representation and recompute the critical path with these precedence relationships.

10. Suppose that the precedence relationships for Problem 2 in Table 10-17 are all direct finish-to-start relationships with no lags except for the following:
    - C to D: S-S with a lag of 1
    - D to E: F-F with a lag of 3
    - A to F: S-S with a lag of 2
    - H to I: F-F with a lag of 4
    - L to M: S-S with a lag of 1

Formulate an activity-on-node network representation and recompute the critical path with these precedence relationships.

11 to 12.

For the projects described in Tables 10-20 and 10-21, respectively, suggest a project schedule that would complete the project in minimum time and result in relatively constant or level requirements for labor over the course of the project.

TABLE 10-20

Activity	A	B	C	D	E	F	G	H	I	J	K
Predecessors	---	---	---	A	B	B	C	C	D,E	F,G	H
Duration	3	5	1	1	7	6	4	3	6	4	3
Workers Per Day	9	6	4	10	16	9	5	8	2	3	7

TABLE 10-21

Activity	A	B	C	D	E	F	G	H	I	J	K	L	M	N
Predecessors	---	---	---	A	A	A	B	B	C	F,G	H,I,L	F,G	D,J	E,K
Duration	5	1	7	2	6	4	3	2	6	4	5	1	4	5
Workers Per Day	0	3	0	9	5	4	2	14	10	4	1	2	7	3

13. Develop a spreadsheet template that lists activity name, duration, required resources, earliest possible start, latest possible start, and scheduled start in separate columns for a maximum of twenty activities. By means of formulas, also develop the required resources for each day of the project, based on the activities' scheduled start, expected durations, and required resources. Use the spreadsheet graphics facility to plot the required resources over time. Use your template to solve Problems 11 and 12 by altering scheduled start times. (Hint: One way to prepare such a template is to use a column to represent a single day with each cell in the column indicating resources required by a particular activity on the particular day).

14. Develop an example of a project network with three critical paths.

15. For the project defined in Table 10-20, suppose that you are limited to a maximum of 20 workers at any given time. Determine a desirable schedule for the project, using the late start time heuristic described in Section 10.9.

16. For the project defined in Table 10-21, suppose that you are limited to a maximum of 15 workers at any given time. Determine a desirable schedule for the project, using the late start time heuristic described in Section 10.9.

17. The examples and problems presented in this chapter generally make use of activity duration and project durations as measured in working days from the beginning of the project. Outline the procedures by which time measured in working days would be converted into calendar days with single- or double-shift work. Could your procedure be modified to allow some but not all activities to be underway on weekends?

Back to top

10.12 Footnotes

1. See Au, T., *Introduction to Systems Engineering, Deterministic Models*, Addison-Wesley Publishing Company, Reading, MA, 1973, for a detailed description of linear programming as a form of mathematical optimization. Back

2. See K.C. Crandall, "Project Planning with Precedence Lead/Lag Factors," *Project Management Quarterly*, Vol. 4, No. 3, Sept. 1973, pp. 18-27, or J.J. Moder, C.R. Phillips, and E.W. Davis, *Project Management with CPM, PERT and Precedence Diagramming*, New York: Van Nostrand Reinhold Company, third edition, 1983, chapter 4. Back

3. See C.T. Hendrickson and B.N. Janson, "A Common Network Formulation of Several Civil Engineering Problems," *Civil Engineering Systems*, Vol. 1, No. 4, 1984, pp. 195-203. Back

4. See IBM, *Project Management System, Application Description Manual*, (H20-0210), IBM, 1968. Back

5. A variety of mathematical programming techniques have been proposed for this problem. For a review and comparison, see J.H. Patterson, "A Comparison of Exact Approaches for Solving the Multiple Constrained Resource Project Scheduling Problem," *Management Science*, Vol. 30, No. 7, 1984, pp. 854-867. Back

6. This example is adapted from H. Smallowitz, "Construction by Computer," *Civil Engineering*, June, 1986, pp. 71-73. Back

7. For discussions and comparisons of alternative heuristic algorithms, see E.M. Davies, "An experimental investigation of resource allocation in multiactivity projects," *Operational Research Quarterly* Vol. 24, No. 11, July 1976, pp. 1186-1194; J.D. Wiest and F.K. Levy, *A Management Guide to PERT/CPM*, Prentice-Hall, New Jersey, 1977; or S.R. Lawrence, *A Computational Comparison of Heuristic Scheduling Techniques*, Technical Report, Graduate School of Industrial Administration, Carnegie-Mellon University, 1985. Back

Previous Chapter | Table of Contents | Next Chapter